aws

Guida per l'utente

# Amazon EKS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

# Amazon EKS: Guida per l’utente

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in qualsiasi modo che possa causare confusione tra i clienti o in qualsiasi modo che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

# Table of Contents


..... xxix
..... xxx
Che cosa è Amazon EKS? ..... 1
Amazon EKS: gestione semplificata di Kubernetes ..... 1
Creazione e scalabilità con Kubernetes: Amazon EKS Capabilities ..... 2
Funzionalità di Amazon EKS ..... 3
Servizi correlati ..... 4
Prezzi di Amazon EKS ..... 5
Casi di utilizzo comune ..... 6
Architecture ..... 7
Piano di controllo ..... 8
Calcolo ..... 8
Funzionalità EKS ..... 10
Concetti di Kubernetes ..... 11
Perché Kubernetes? ..... 12
Cluster ..... 16
Carichi di lavoro ..... 21
Fasi successive ..... 27
Opzioni di implementazione ..... 28
Comprensione delle opzioni di implementazione di Amazon EKS ..... 28
Amazon EKS nel cloud ..... 28
Amazon EKS nei tuoi data center o ambienti edge ..... 29
Amazon EKS Anywhere per ambienti isolati ..... 30
Strumenti di Amazon EKS ..... 31
Configurazione ..... 32
Fasi successive ..... 32
Configurare la AWS CLI ..... 32
Per creare una chiave di accesso ..... 33
Per configurare AWS CLI ..... 33
Per ottenere un token di sicurezza ..... 34
Per verificare l'identità dell'utente ..... 34
Fasi successive ..... 35
Configura kubectl ed eksctl ..... 35
`Installa o aggiorna kubectl` ..... 35

Passaggio 1: controlla se kubectl è installato ... 36
Passaggio 2: installa o aggiorna kubectl ... 36
Installa `eksctl` ... 48
Fasi successive ... 49
Guida introduttiva ... 50
In questo tutorial ... 50
Prerequisiti ... 50
Configurazione del cluster ... 51
Crea IngressClass ... 51
Distribuzione dell'applicazione di gioco 2048 esemplificativa ... 52
Dati persistenti utilizzando la modalità automatica Amazon EKS ... 55
Eliminazione ... 57
Scopri Amazon EKS ... 58
Panoramica di ... 58
Workshop su Amazon EKS ... 58
Tutorial pratici per la configurazione dei cluster su Amazon EKS ... 60
Esempi di Amazon EKS ... 61
AWS Esercitazioni ... 61
Workshop per sviluppatori ... 62
Workshop Terraform ... 62
AWS Formazione Amazon EKS ... 62
Nozioni di base ... 63
Crea cluster (EKS Auto Mode) ... 63
Crea cluster (eksctl) ... 64
Prerequisiti ... 65
Passaggio 1: Creazione di cluster e nodi Amazon EKS ... 65
Passaggio 2: Visualizzazione delle risorse Kubernetes ... 66
Passaggio 3: Eliminazione del cluster e dei nodi ... 68
Fasi successive ... 68
Crea cluster (console e CLI) ... 69
Prerequisiti ... 69
Passaggio 1: Creazione del cluster Amazon EKS; ... 70
Fase 2: Connettiti al tuo cluster ... 73
Passaggio 3: Creazione di nodi ... 74
Passaggio 4: Visualizzazione delle risorse ... 76
Passaggio 5. Eliminazione delle risorse ... 77

Fasi successive ..... 78
Modalità automatica EKS ..... 79
Funzionalità ..... 79
Componenti automatici ..... 81
Configurazione ..... 82
Modello di responsabilità condivisa ..... 83
Creazione di un cluster ..... 84
CLI di eksctl ..... 85
AWS CLI ..... 87
Console di gestione ..... 96
Abilita i cluster esistenti ..... 98
Materiale di riferimento per le migrazioni ..... 99
Migration di volumi EBS ..... 99
Migrazione dei bilanciatori del carico ..... 100
Abilitazione sul cluster ..... 101
Eseguire la migrazione da Karpenter ..... 105
Eseguire la migrazione da MNG ..... 109
Eseguire la migrazione da Fargate ..... 109
Esegui carichi di lavoro ..... 117
Implementazione di un carico di lavoro di decompressione ..... 118
Implementa il bilanciatore del carico ..... 121
Implementazione di un carico di lavoro stateful ..... 126
Implementare carico di lavoro accelerato ..... 132
Configura ..... 139
Creazione di una classe di nodi ..... 144
Creazione di pool di nodi ..... 160
Crea un pool di nodi statico ..... 167
Creare una classe in ingresso ..... 174
Creare servizio ..... 183
Multi-cluster gruppi target ..... 189
Condividi un gruppo target ..... 191
Concedi RBAC ai controller gestiti ..... 193
Concedi l'accesso segreto a LBC ..... 195
Crea StorageClass ..... 198
Disabilita modalità automatica di EKS ..... 208
Aggiornamento della versione di Kubernetes ..... 209

Riesamina i pool di nodi integrati ..... 211
Controllare l'implementazione ..... 213
Eseguire componenti aggiuntivi critici ..... 216
Usare le policy di rete ..... 218
Assegnazione di tag alle sottoreti ..... 230
Generare report CIS. ..... 232
Crittografare i volumi root ..... 235
Aggiorna i controlli AMI ..... 239
Controllare le prenotazioni della capacità ..... 242
Usa Local Zones ..... 249
Sicurezza avanzata ..... 253
Ottimizzazione dei costi ..... 255
Come funziona ..... 264
Istanze gestite ..... 264
Identità e accesso ..... 269
Rete ..... 272
Osservabilità ..... 276
Registri dei componenti gestiti automaticamente da EKS ..... 277
Risoluzione dei problemi ..... 279
Agente di monitoraggio del nodo ..... 280
Ottieni l'output della console da un'istanza gestita da EC2 utilizzando AWS CLI EC2 ..... 280
Ottieni i log dei nodi usando contenitori di debug e la CLI di `kubectl` ..... 281
Visualizza le risorse associate a EKS Auto Mode nel AWS Console ..... 282
Visualizza gli errori IAM nel tuo AWS account ..... 283
Risolvere i problemi relativi alla mancata pianificazione dei Pod sul nodo Auto Mode ..... 284
Risolvere i problemi del nodo che non si unisce al cluster ..... 284
Condivisione di volumi tra pod ..... 287
Visualizzare gli eventi di Karpenter nei log del piano di controllo ..... 287
Risolvere i problemi relativi ai controller inclusi in Auto Mode ..... 289
Risorse correlate ..... 198
Note di rilascio ..... 289
17 agosto 2026 ..... 290
12 agosto 2026 ..... 290
5 agosto 2026 ..... 291
27 luglio 2026 ..... 291
21 luglio 2026 ..... 292

30 giugno 2026 ..... 292
9 giugno 2026 ..... 294
4 giugno 2026 ..... 294
3 giugno 2026 ..... 294
5 maggio 2026 ..... 294
10 aprile 2026 ..... 294
2 aprile 2026 ..... 295
2 febbraio 2026 ..... 295
19 dicembre 2025 ..... 295
19 novembre 2025 ..... 295
19 novembre 2025 ..... 295
23 ottobre 2025 ..... 295
1° ottobre 2025 ..... 296
30 settembre 2025 ..... 296
29 settembre 2025 ..... 296
10 settembre 2025 ..... 296
24 agosto 2025 ..... 296
15 agosto 2025 ..... 296
6 agosto 2025 ..... 296
30 giugno 2025 ..... 296
20 giugno 2025 ..... 297
13 giugno 2025 ..... 297
30 aprile 2025 ..... 297
18 aprile 2025 ..... 297
11 aprile 2025 ..... 297
4 aprile 2025 ..... 298
31 marzo 2025 ..... 298
21 marzo 2025 ..... 298
14 marzo 2025 ..... 298
Configurazione dei cluster ..... 299
Creare un cluster automatico ..... 300
Prerequisiti ..... 50
Crea cluster - AWS console ..... 301
Crea cluster - AWS CLI ..... 306
Fasi successive ..... 96
Creazione di un cluster ..... 312

Prerequisiti ..... 50
Passaggio 1: crea un ruolo IAM del cluster ..... 313
Fase 2: Creare un cluster ..... 314
Passaggio 3: aggiornamento di kubeconfig ..... 324
Passaggio 4: configurazione del cluster ..... 325
Fasi successive ..... 96
Crea un piano di controllo predisposto ..... 326
Panoramica di ..... 93
Casi d'uso ..... 153
Livelli di scalabilità del piano di controllo ..... 329
Considerazioni ..... 100
Guida introduttiva a Provisioned Control Plane ..... 336
Configurazione avanzata del piano di controllo ..... 344
Panoramica di ..... 93
Parametri del piano di controllo Kubernetes supportati ..... 345
Considerazioni ..... 100
Fasi successive ..... 96
Configura i parametri del piano di controllo ..... 356
Approfondimenti sui cluster ..... 367
Tipi di approfondimenti del cluster ..... 367
Considerazioni ..... 367
Casi d'uso ..... 368
Nozioni di base ..... 370
Visualizzazione degli approfondimenti sui cluster ..... 370
Aggiornamento della versione di Kubernetes ..... 380
Considerazioni per la modalità automatica di Amazon EKS ..... 190
Riepilogo ..... 382
Passaggio 1: preparazione per l'aggiornamento ..... 382
Passaggio 2: esaminare le considerazioni sull'aggiornamento ..... 383
Fase 3: aggiornamento del piano di controllo del cluster ..... 385
Fase 4: aggiornamento dei componenti del cluster ..... 387
Effettuare il downgrade della versione di Kubernetes di un cluster Amazon EKS ..... 388
Ripristina la versione di Kubernetes ..... 389
Cosa viene ripristinato ..... 389
Cosa NON viene ripristinato ..... 389
Prerequisiti ..... 390

Riepilogo ..... 391
Fase 1: Rivedi le informazioni sulla preparazione al rollback ..... 392
Fase 2: Preparazione dei nodi di lavoro ..... 394
Fase 3: Ripristina il piano di controllo del cluster ..... 395
Fase 4: Monitora l'avanzamento del rollback ..... 397
Considerazioni e avvertenze ..... 398
Rollback e componenti aggiuntivi ..... 399
Risorse correlate ..... 400
Ripristina i cluster EKS Auto Mode ..... 400
Eliminazione di un cluster ..... 408
Considerazioni ..... 100
Prerequisiti ..... 409
Eliminare un cluster (eksctl) ..... 410
Elimina cluster (AWS console) ..... 410
Elimina cluster (AWS CLI) ..... 412
Proteggi i cluster EKS dall'eliminazione accidentale ..... 413
Accesso all'endpoint del cluster ..... 414
`Formato endpoint del cluster` IPv6 ..... 415
`Formato endpoint del cluster IPv4` ..... 415
Endpoint privato del cluster ..... 416
Modifica dell'accesso all'endpoint del cluster ..... 417
Accesso a un server API solo privato ..... 420
Configurare l'accesso all'endpoint ..... 422
Abilitazione del supporto Windows ..... 425
Considerazioni ..... 100
Prerequisiti ..... 50
Abilitazione del supporto Windows ..... 427
Implementazione di pods Windows ..... 429
Supporto di maggiore densità di pod sui nodi di Windows ..... 430
Disabilitare il supporto Windows ..... 430
Cluster privati ..... 431
Requisiti dell'architettura del cluster ..... 431
Requisiti dei nodi ..... 432
Requisiti del pod ..... 433
Uscita del piano di controllo ..... 436
Modalità di routing in uscita ..... 437

Prerequisiti ..... 438
Crea un cluster con i dati di uscita indirizzati dal cliente ..... 440
Aggiornare un cluster esistente ..... 441
Considerazioni sull'IPv6 ..... 442
Considerazioni ..... 442
Chiave di condizione IAM ..... 443
Configurazione del provider OIDC ..... 444
Verifica la connettività ..... 444
Risolvi i problemi relativi all'uscita ..... 445
Dimensionamento automatico ..... 452
EKS Auto Mode ..... 452
Ulteriori soluzioni ..... 453
Ulteriori informazioni relative allo spostamento zonale ..... 454
Comprensione del flusso di traffico di rete est-ovest tra i pod ..... 454
Requisiti per lo spostamento zonale EKS ..... 459
Domande frequenti ..... 468
Risorse aggiuntive ..... 471
Abilitazione dello spostamento zonale ..... 471
Considerazioni ..... 472
What is Amazon Application Recovery Controller? ..... 472
Che cos'è uno spostamento zonale? ..... 473
Cos'è lo spostamento automatico zonale automatico? ..... 473
Cosa fa EKS durante un cambio di zona? ..... 473
Registra il cluster EKS con Amazon Application Recovery Controller (ARC) (AWS console) ..... 474
Fasi successive ..... 96
Gestione dell'accesso ..... 475
Attività comuni ..... 475
Contesto ..... 230
Considerazioni relative alla modalità automatica di EKS ..... 409
Accesso API Kubernetes ..... 476
Associa le identità IAM alle autorizzazioni Kubernetes ..... 477
Set Cluster Authentication Mode ..... 478
Voci di accesso ..... 480
aws-auth ConfigMap ..... 538
Collegare il provider OIDC ..... 550

Scollegare il gestore OIDC ..... 556
Accesso alle risorse di cluster ..... 556
Autorizzazioni richieste ..... 558
CloudTrail visibilità ..... 561
Modifica le risorse del cluster ..... 562
Autorizzazioni richieste ..... 562
CloudTrail visibilità ..... 561
Accedere al cluster con kubectl ..... 563
Connettiti usando la console EKS () CloudShell ..... 564
`Crea il` file kubeconfig usando AWS CLI ..... 565
Accesso al carico di lavoro a AWS ..... 566
Token dell'account di servizio ..... 566
Componenti aggiuntivi del cluster ..... 568
Concessione AWS Autorizzazioni di Identity and Access Management per i carichi di lavoro sui cluster Amazon Elastic Kubernetes Service ..... 568
Credenziali con IRSA ..... 573
Pod Identity ..... 599
Gestione del calcolo ..... 635
Confronto delle opzioni di calcolo ..... 635
Gruppi di nodi gestiti ..... 643
Concetti sui gruppi di nodi gestiti ..... 644
Tipi di capacità del gruppo di nodi gestiti ..... 646
Crea ..... 650
Aggiorna ..... 665
Dettagli sull'aggiornamento del comportamento ..... 671
Modelli di avvio ..... 675
Elimina ..... 692
Nodi autogestiti ..... 694
Amazon Linux ..... 696
Razzo per bottiglie ..... 705
Windows ..... 709
Ubuntu Linux ..... 718
Metodi di aggiornamento ..... 721
AWS Fargate ..... 734
AWS Considerazioni su Fargate ..... 735
Tabella di confronto Fargate ..... 738

Nozioni di base ... 740
Definisci i profili ... 746
Eliminare profili ... 751
Dettagli di configurazione del pod ... 753
Eventi di applicazione di patch del sistema operativo ... 756
Raccolta dei parametri ... 758
Registrazione dei log ... 761
Tipi di istanza Amazon EC2 ... 772
Come viene determinato MaxPods ... 774
Considerazioni per la modalità automatica di EKS ... 409
AMI ottimizzate preconfigurate ... 776
Definizione delle AMI AL2 come obsolete ... 776
Amazon Linux ... 783
Portabottiglie ... 793
Ubuntu Linux ... 809
Windows ... 810
Salute e riparazione dei nodi ... 900
Agente di monitoraggio del nodo ... 901
Riparazione automatica dei nodi ... 902
Rilevamento dello stato dei nodi ... 903
Riparazione automatica dei nodi ... 923
Visualizzazione dello stato dei nodi ... 929
Ottenere i log dei nodi ... 931
Cattura il traffico di rete ... 935
Nodi ibridi ... 938
Funzionalità ... 939
Limits ... 940
Considerazioni ... 940
Risorse aggiuntive ... 941
Prerequisiti ... 941
Esegui i nodi ibridi ... 1011
Configura ... 1033
Come funziona ... 1120
nodeadm dei nodi ibridi ... 1161
Risoluzione dei problemi ... 1179
Gateway di nodi ibridi ... 1201

Archiviazione di dati per app ..... 1243
Amazon EBS ..... 1243
Considerazioni ..... 1244
Prerequisiti ..... 1244
Passaggio 1: Creazione di un ruolo IAM ..... 1245
Fase 2: ottenere il driver CSI di Amazon EBS ..... 1253
Fase 3: implementazione di un'applicazione di esempio ..... 1254
Archivio dell'istanza Amazon EC2 ..... 1254
Considerazioni ..... 1254
Prerequisiti ..... 1255
Installazione del driver CSI di Amazon EC2 Instance Store ..... 1255
Implementazione di un'applicazione di esempio ..... 1256
I/O strozzamento ..... 1256
Modifiche allo spazio dei nomi NVMe sulle istanze appena lanciate ..... 1258
Amazon EFS ..... 1259
Considerazioni ..... 1259
Prerequisiti ..... 1260
Passaggio 1: Creazione di un ruolo IAM ..... 1261
Fase 2: ottenere il driver CSI per Amazon EFS ..... 1267
Fase 3: Creazione di un file system Amazon EFS ..... 1268
Fase 4: implementazione di un'applicazione di esempio ..... 1268
File di Amazon S3 ..... 1268
Considerazioni ..... 1268
Prerequisiti ..... 1268
Fase 1: Creare ruoli IAM ..... 1269
Fase 2: ottenere il driver CSI per Amazon EFS ..... 1279
Fase 3: creare un file system Amazon S3 ..... 1279
Fase 4: implementazione di un'applicazione di esempio ..... 1279
Amazon FSx per Lustre ..... 1280
Implementare il driver ..... 1280
Ottimizza (EFA) ..... 1287
Ottimizza (non EFA) ..... 1304
Amazon FSx per ONTAP NetApp ..... 1311
Amazon FSx per OpenZFS ..... 1312
Amazon File Cache ..... 1313
Mountpoint per Amazon S3 ..... 1313

Considerazioni ..... 1313
Implementazione del driver ..... 1314
Rimozione del driver ..... 1324
Controller snapshot CSI ..... 1326
Configurazione delle reti ..... 1328
Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione . 1328
Requisiti del VPC e delle sottoreti ..... 1328
Considerazioni e requisiti relativi al VPC ..... 1329
Considerazioni e requisiti relativi alle sottoreti ..... 1331
Considerazioni e requisiti relativi alle sottoreti condivisi ..... 1338
Crea un VPC ..... 1339
Prerequisiti ..... 50
Sottoreti pubbliche e private ..... 1339
Solo sottoreti pubbliche ..... 1342
Solo sottoreti private ..... 1344
Requisiti relativi al gruppo di sicurezza ..... 1346
Gruppo di sicurezza del cluster predefinito ..... 1346
Limitazione del traffico cluster ..... 1347
Gruppi di sicurezza condivisi ..... 1349
Gestione di componenti aggiuntivi di rete ..... 1350
Built-in componenti aggiuntivi ..... 1350
Facoltativo AWS componenti aggiuntivi di rete ..... 1351
CNI di Amazon VPC ..... 1351
Plug-in CNI alternativi ..... 1480
Multus ..... 1482
AWS Controller Load Balancer ..... 1483
CoreDNS ..... 1502
kube-proxy ..... 1520
Carichi di lavoro ..... 1526
Implementazione esemplificativa (Linux) ..... 1526
Prerequisiti ..... 50
Creare un namespace ..... 1527
Creazione di un'implementazione Kubernetes ..... 1527
Creazione di un servizio ..... 1529
Esame delle risorse create ..... 1530
Eseguire una shell su un pod ..... 1533

Fasi successive ..... 1534
Implementazione esemplificativa (Windows) ..... 1534
Prerequisiti ..... 50
Creare un namespace ..... 1527
Creazione di un'implementazione Kubernetes ..... 1527
Creazione di un servizio ..... 1529
Esame delle risorse create ..... 1537
Eseguire una shell su un pod ..... 1533
Fasi successive ..... 1541
Vertical Pod Autoscaler ..... 1541
Implementazione di Vertical Pod Autoscaler ..... 1542
Test dell'installazione di Vertical Pod Autoscaler ..... 1543
Horizontal Pod Autoscaler ..... 1547
Esecuzione di un'applicazione di test di Horizontal Pod Autoscaler ..... 1548
Bilanciamento del carico di rete ..... 1551
Prerequisiti ..... 50
Considerazioni ..... 100
Creazione di un Network Load Balancer ..... 1555
(Facoltativo) Implementare un'applicazione di esempio ..... 1557
Bilanciamento del carico di applicazione ..... 1561
Prerequisiti ..... 50
Riutilizzare ALB con gruppi di ingresso ..... 1564
(Facoltativo) Implementare un'applicazione di esempio ..... 1565
Limitazione degli IP esterni ai servizi ..... 1568
Copia di un'immagine in un repository ..... 1571
Visualizza i registri delle immagini di Amazon ..... 1574
Componenti aggiuntivi Amazon EKS ..... 1578
Considerazioni ..... 1579
Namespace personalizzato per componenti aggiuntivi ..... 1580
Considerazioni per la modalità automatica di Amazon EKS ..... 1580
Supporto ..... 1581
AWS componenti aggiuntivi ..... 1584
Componenti aggiuntivi della community ..... 1608
Componenti aggiuntivi di Marketplace ..... 1614
Creare un componente aggiuntivo ..... 1633
Aggiornamento di un componente aggiuntivo ..... 1646

Verifica la compatibilità ... 1654
Rimozione di un componente aggiuntivo ... 1656
Ruoli IAM ... 1660
Campi che puoi personalizzare ... 1667
Verifica delle immagini di container ... 1670
Funzionalità EKS ... 1672
Funzionalità disponibili ... 1672
AWS Controller per Kubernetes (ACK) ... 1672
Argo CD ... 1673
kro (Kube Resource Orchestrator) ... 1673
Vantaggi delle funzionalità EKS ... 1674
Prezzi ... 278
Come funzionano le funzionalità EKS ... 1675
Casi di utilizzo comune ... 1676
Lavorare con le funzionalità ... 1677
Risorse relative alle funzionalità EKS ... 1677
Comprensione dello stato della capacità ... 1678
Crea funzionalità ... 1679
Elenca le funzionalità ... 1680
Descrivi una funzionalità ... 1681
Aggiorna la configurazione di una funzionalità ... 1682
Eliminare una funzionalità ... 1683
Fasi successive ... 96
Funzionalità, registri dei controller ... 1684
Tipi di log supportati ... 1685
Impostazione della consegna dei log ... 277
Esempio: interrogazione dei log con Logs Insights CloudWatch ... 1687
Visualizzazione dei log ... 278
Prezzi ... 278
risorse Kubernetes ... 1688
Risorse Argo CD ... 1688
risorse kro ... 1689
Risorse ACK ... 1690
Limiti delle risorse ... 1691
Fasi successive ... 96
Considerazioni ... 1692

Capacità, ruoli IAM e Kubernetes RBAC ..... 1692
Multi-cluster modelli di architettura ..... 1693
Confronto tra le funzionalità EKS e le soluzioni autogestite ..... 1695
AWS Controller per Kubernetes ..... 1698
Come funziona ACK ..... 1698
Vantaggi dell'ACK ..... 1699
Supportata AWS services ..... 1700
Integrazione con altre funzionalità gestite da EKS ..... 1700
Guida introduttiva a ACK ..... 1701
Crea funzionalità ACK ..... 1701
Concetti ACK ..... 1711
Configurazione delle autorizzazioni ..... 1718
Considerazioni ..... 1726
Risoluzione dei problemi ..... 1734
Confronto con l'autogestito ..... 1738
Argo CD ..... 1742
Come funziona Argo CD ..... 1742
Vantaggi di Argo CD ..... 1744
Integrazione con AWS Centro identità ..... 1744
Integrazione con altre funzionalità gestite da EKS ..... 1700
Guida introduttiva ad Argo CD ..... 1745
Crea una funzionalità Argo CD ..... 1745
Concetti di Argo CD ..... 1761
Configurazione delle autorizzazioni ..... 1768
Lavorare con Argo CD ..... 1781
Accesso privato agli endpoint ..... 1838
Considerazioni su Argo CD ..... 1843
Risoluzione dei problemi ..... 1852
Confronto con l'autogestito ..... 1865
kro ..... 1870
Come funziona kro ..... 1871
Vantaggi di kro ..... 1871
Integrazione con altre funzionalità gestite da EKS ..... 1700
Iniziare con kro ..... 1873
Crea una capacità kro ..... 1873
concetti kro ..... 1885

Configurazione delle autorizzazioni ... 1892
Considerazioni ... 1897
Risoluzione dei problemi ... 1903
Confronto con quello autogestito ... 1909
Funzionalità di risoluzione dei problemi ... 1910
Approccio generale alla risoluzione dei problemi ... 1911
Usa i log del controller per la risoluzione dei problemi ... 1911
Verifica lo stato della funzionalità ... 1913
Stati di capacità comuni ... 1913
Verifica lo stato delle risorse Kubernetes ... 1914
Rivedi le autorizzazioni IAM e l'accesso al cluster ... 1915
Capability-specific risoluzione dei problemi ... 1917
Problemi comuni a tutte le funzionalità ... 1917
Fasi successive ... 96
Gestione dei cluster ... 1919
Monitoraggio dei costi ... 1919
Visualizzare i costi per pod ... 1920
Installare Kubecost ... 1921
Accedere alla dashboard Kubecost ... 1923
Informazioni su Kubecost ... 1924
Metrics Server ... 1936
Considerazioni ... 100
Implementa come componente aggiuntivo della community con Amazon EKS Add-ons ... 1937
Implementa con manifesto ... 1938
Implementazione di app con Helm ... 1939
Tagging delle risorse ... 1941
Nozioni di base sui tag ... 1941
Tagging delle risorse ... 1942
Limitazioni applicate ai tag ... 1943
Tagging delle risorse per la fatturazione ... 1943
Utilizzo di tag tramite la console ... 1944
Lavorare con i tag utilizzando la CLI, l'API o eksctl ... 1945
Service Quotas ... 1947
Visualizza le quote dei servizi EKS nel Console di gestione AWS ... 1947
Visualizza le quote dei servizi EKS con la CLI AWS ... 1947
Service Quotas di Amazon EKS ... 1948

AWS Quote di servizio Fargate ... 1949
Sicurezza ... 1950
Best practice ... 1951
Analizzare le vulnerabilità ... 1951
Il Center for Internet Security (CIS) Benchmark per Amazon EKS ... 1952
Versioni della piattaforma Amazon EKS ... 1952
Elenco delle vulnerabilità del sistema operativo ... 1952
Rilevamento dei nodi con Amazon Inspector ... 1953
Rilevamento di cluster e nodi con Amazon GuardDuty ... 1953
Convalida della conformità ... 1953
Considerazioni per EKS ... 1954
Sicurezza dell'infrastruttura ... 1955
Resilienza ... 1961
Prevenzione del problema "confused deputy" tra servizi ... 1962
Considerazioni su Kubernetes ... 1964
Firma dei certificati ... 1965
Rotazione CA ... 1967
Ruoli e utenti predefiniti ... 2005
Abilitare la crittografia dei segreti ... 2010
AWS Secrets Manager ... 2014
Crittografia a busta predefinita per tutti i dati dell'API Kubernetes ... 2015
Indurimento RBAC ... 2023
Considerazioni per EKS Auto ... 2026
Sicurezza e autenticazione delle API ... 2026
Sicurezza di rete ... 2026
Sicurezza delle istanze gestite di EC2 ... 2027
Gestione automatizzata delle risorse ... 2028
Best practice di sicurezza ... 2029
Considerazioni sulle funzionalità EKS ... 2029
Capability: ruolo IAM ... 2030
Voci di accesso EKS ... 2030
Autorizzazioni Kubernetes aggiuntive ... 2032
Autorizzazioni IAM richieste dalla funzionalità ... 2034
Best practice di sicurezza ... 2034
Fasi successive ... 96
Riferimento IAM ... 2038

Destinatari ..... 2038
Autenticazione con identità ..... 2038
Gestione dell'accesso con policy ..... 2042
Amazon EKS e IAM ..... 2044
Identity-based politiche ..... 2049
Ruoli collegati ai servizi ..... 2057
Ruolo IAM per l'esecuzione del pod ..... 2074
Ruolo IAM di Connector ..... 2079
AWS politiche gestite ..... 2083
Risoluzione dei problemi ..... 2121
Ruolo IAM del cluster ..... 2124
Ruolo IAM del nodo ..... 2127
Ruolo IAM del cluster Auto Mode ..... 2133
Ruolo IAM del nodo in Auto Mode ..... 2140
Capacità, ruolo IAM ..... 2143
Monitoraggio dei cluster ..... 2149
Registrazione e monitoraggio su Amazon EKS ..... 2149
Strumenti di registrazione e monitoraggio di Amazon EKS ..... 2151
Dashboard di osservabilità ..... 2155
Riepilogo ..... 2155
Integrità del cluster ..... 2155
Monitoraggio del piano di controllo ..... 2155
Approfondimenti sui cluster ..... 2158
Problemi di integrità dei nodi ..... 2158
Funzionalità EKS ..... 2158
Osservabilità della rete di container ..... 2159
Casi d'uso ..... 153
Funzionalità ..... 79
Nozioni di base ..... 370
Prerequisiti e note importanti ..... 2161
Utilizzo di AWS CLI, API EKS e API NFM ..... 2162
Utilizzo dell'infrastruttura come codice (IaC) ..... 2164
Come funziona? ..... 222
Considerazioni e limitazioni ..... 1998
Parametri Prometheus ..... 2175
Passaggio 1: attivazione delle metriche Prometheus ..... 2176

Passaggio 2: utilizzo delle metriche Prometheus ..... 2178
Passaggio 3: gestione degli scraper Prometheus ..... 2178
Implementare con Helm ..... 2178
Piano di controllo ..... 2182
Amazon CloudWatch ..... 2191
Metriche di base in Amazon CloudWatch ..... 2191
Operatore di CloudWatch osservabilità di Amazon ..... 2201
Registrazione dei log del piano di controllo (control-plane) ..... 2202
Abilitare o disabilitare i log del piano di controllo (control-plane) ..... 2203
Visualizzare i log del piano di controllo (control-plane) del cluster ..... 2206
AWS CloudTrail ..... 2207
Riferimenti ..... 2208
Voci del file di log ..... 2208
Parametri del gruppo con scalabilità automatica ..... 2211
Operatore ADOT ..... 2212
Pannello di controllo Amazon EKS ..... 2213
Cos'è il pannello di controllo Amazon EKS? ..... 2213
Come viene utilizzata la dashboard AWS Organizzazioni? ..... 2214
Abilita l'EKS Dashboard utilizzando il AWS console ..... 2215
Visualizzazione del pannello di controllo EKS ..... 2216
Configura il pannello di controllo ..... 2217
Disabilita il pannello di controllo EKS usando il AWS console ..... 2218
Risoluzione dei problemi del pannello di controllo EKS ..... 2218
Uso di altri servizi ..... 2225
AWS Backup ..... 2225
AWS CloudFormation ..... 2226
Modelli Amazon EKS e AWS CloudFormation ..... 2226
Ulteriori informazioni su AWS CloudFormation ..... 2227
AWS CodeConnections ..... 2227
Utilizzare AWS CodeConnections con Argo CD ..... 2227
Considerazioni sull'utilizzo con Argo CD CodeConnections ..... 2228
Amazon Detective ..... 2229
Uso di Amazon Detective con Amazon EKS ..... 2229
Amazon GuardDuty ..... 2230
AWS Zone locali ..... 2231
AWS Resilience Hub ..... 2232

AWS Secrets Manager ..... 2232
Usare AWS Secrets Manager con Argo CD ..... 2232
Amazon Security Lake ..... 2235
Vantaggi dell'utilizzo di Security Lake con Amazon EKS ..... 2235
Abilitare Security Lake per Amazon EKS ..... 2235
Analisi dei log EKS in Security Lake ..... 2236
Amazon VPC Lattice ..... 2236
Amazon EKS Connector ..... 2237
Considerazioni su Amazon EKS Connector ..... 2237
Ruoli IAM richiesti per Amazon EKS Connector ..... 2238
Connettere un cluster ..... 2238
Considerazioni ..... 2239
Prerequisiti ..... 2239
Fase 1: Registrazione del cluster ..... 2239
Fase 2: Installazione dell'agente `eks-connector` ..... 2242
Fasi successive ..... 2244
Garantire l'accesso ai cluster ..... 2244
Prerequisiti ..... 2244
Annullamento della registrazione di un cluster ..... 2246
Annullare la registrazione del cluster Kubernetes ..... 2246
Eliminare le risorse del cluster Kubernetes ..... 2247
Risoluzione dei problemi del connettore EKS ..... 2248
Risoluzione dei problemi di base ..... 2248
Problema di Helm: 403 Forbidden ..... 2250
Errore della console: il cluster è bloccato nello stato Pending (In sospeso) ..... 2250
Errore della console: User system:serviceaccount:eks-connector:eks-connector non può impersonare la risorsa utenti nel gruppo API nell'ambito del cluster ..... 2251
Errore della console: […] non consentito: L'utente […] non può elencare la risorsa […] nel gruppo API nell'ambito del cluster ..... 2251
Errore della console: Amazon EKS non è in grado di comunicare con il server API cluster Kubernetes. Per una connessione corretta, il cluster deve trovarsi nello stato ACTIVE (ATTIVO). Riprova tra qualche minuto. ..... 2252
I Connector Pod di Amazon EKS si bloccano in modo anomalo ..... 2252
Avvio di eks-connector non riuscito: InvalidActivation ..... 2253
Nel nodo del cluster manca la connettività in uscita ..... 2254
I Connector Pod di Amazon EKS sono in stato `ImagePullBackOff` ..... 2254

Domande frequenti ..... 2255
Considerazioni relative alla sicurezza ..... 2256
Responsabilità di AWS ..... 2256
Responsabilità del cliente ..... 2256
Amazon EKS su AWS Outposts ..... 2258
Quando utilizzare ciascuna opzione di implementazione ..... 2258
Due implementazioni di cluster locali ..... 2259
Confronto tra le opzioni di implementazione ..... 2259
Esecuzione di cluster locali ..... 2264
Supportata AWS Regioni ..... 2265
Implementazione di un cluster locale ..... 2265
Versioni della piattaforma EKS ..... 2276
Creazione di un VPC e delle sottoreti ..... 2290
Preparazione per le disconnessioni ..... 2294
Considerazioni sulla capacità ..... 2299
Risoluzione dei problemi dei cluster ..... 2301
Esegui cluster locali (archivio di istanze EC2) ..... 2313
Cluster locali Amazon EKS con archivio di istanze EC2 Outposts ..... 2314
Nozioni di base ..... 2315
Implementa un cluster locale (store di istanze EC2) ..... 2315
Crea un VPC e delle sottoreti (store di istanze EC2) ..... 2321
Considerazioni sulla capacità (EC2 instance store) ..... 2323
Preparati alle disconnessioni (archivio di istanze EC2) ..... 2326
Componenti aggiuntivi di Amazon EKS (archivio di istanze EC2) ..... 2333
Risolvi i problemi relativi ai cluster (store di istanze EC2) ..... 2334
Nodi ..... 2336
`eksctl` ..... 2338
Console di gestione AWS ..... 2340
AI/ML su EKS ..... 2347
Perché usare Amazon EKS per AI/ML ..... 2347
Casi d'uso principali ..... 2348
Casi di studio ..... 2348
Struttura della guida ..... 2349
Inizia a usare AI/ML su Amazon EKS ..... 2349
Configura il cluster ..... 2350
Panoramica dell'architettura ..... 2350

Opzioni di elaborazione in cluster ..... 2351
Cosa configurerai ..... 2351
Nozioni di base ..... 370
Command-line interfaccia (CLI) ..... 2352
Terraform ..... 2403
Inferenza ..... 2428
Perché usare Amazon EKS per l'inferenza ..... 2428
Glossario ..... 2429
Come funziona l'inferenza su Amazon EKS ..... 2429
Metriche delle prestazioni di inferenza ..... 2430
Riduzione del tempo di avvio a freddo ..... 2431
Cosa implementerai ..... 2431
Modello Load & Serve ..... 2432
Dimensionamento automatico ..... 2448
Accelera il caricamento dei modelli ..... 2470
Configurazione del cluster ..... 2483
Usa le AMI GPU EKS Linux ..... 2483
Configura le AMI GPU Windows ..... 2489
Configurazione dei cluster di formazione con EFA ..... 2495
Configura cluster di inferenza con Inferentia ..... 2506
Gestione della capacità ..... 2513
Scegli tra i tipi di istanze EC2 ..... 2513
Scopri le opzioni di acquisto di EC2 ..... 2513
Abbina le opzioni di acquisto ai requisiti del carico di lavoro ..... 2514
Verifica le quote dei servizi EC2 ..... 2514
Usa le opzioni di acquisto EC2 con Amazon EKS ..... 2515
Strategia mista: combina le opzioni di acquisto ..... 2516
EKS Auto Mode e Karpenter ..... 2517
Gruppi di nodi gestiti e nodi autogestiti ..... 2537
Usa P6e-GB200 con EKS ..... 2543
Gestione dei dispositivi ..... 2557
Allocazione dinamica delle risorse rispetto ai plug-in dei dispositivi ..... 2557
Driver DRA per Amazon EKS ..... 2558
Plugin di dispositivo per Amazon EKS ..... 2559
Considerazioni ..... 100
ResourceClaim DRA vs. ResourceClaimTemplate ..... 2560

Argomenti ..... 2562
Dispositivi EFA ..... 2562
Acceleratori Neuron ..... 2579
GPU NVIDIA ..... 2587
Ricette ..... 2633
Impedire che i pod siano pianificati su nodi specifici ..... 2633
Resources ..... 2635
Workshop ..... 2635
Best practice ..... 2636
Architetture di riferimento ..... 2637
Esercitazioni ..... 2638
Tools (Strumenti) ..... 2640
Model Context Protocol (MCP) ..... 2640
Panoramica di ..... 93
Esempi di integrazione ..... 2642
Nozioni di base ..... 370
Nozioni di base ..... 2643
Configurazione ..... 2659
Tools (Strumenti) ..... 2661
Amazon Q ..... 2672
Come funziona ..... 153
Utilizzo di Amazon Q per la risoluzione dei problemi ..... 2672
Considerazioni ..... 100
Ulteriori informazioni ..... 2673
Controllo delle versioni ..... 2674
Versioni di Kubernetes ..... 2674
Versioni disponibili con supporto standard ..... 2675
Versioni disponibili con supporto esteso ..... 2675
Rollback della versione ..... 2676
Calendario di rilascio di Amazon EKS Kubernetes ..... 2676
Ottieni informazioni sulla versione con AWS CLI ..... 2677
Domande frequenti sulle versioni di Amazon EKS ..... 2678
Amazon EKS extended support FAQs ..... 2681
Versioni di supporto standard ..... 2683
Versioni di supporto esteso ..... 2689
Visualizzazione del periodo di supporto ..... 2694

Visualizzazione della policy di aggiornamento ... 2694
Abilitare il supporto esteso ... 2697
Disabilitare il supporto esteso ... 2698
Versioni della piattaforma ... 2699
Kubernetes versione 1.36 ... 2701
Kubernetes versione 1.35 ... 2702
Kubernetes versione 1.34 ... 2705
Kubernetes versione 1.33 ... 2710
Kubernetes versione 1.32 ... 2716
Kubernetes versione 1.31 ... 2724
Kubernetes versione 1.30 ... 2732
Ottieni versione corrente della piattaforma ... 2742
Modificare la versione della piattaforma ... 2743
Risoluzione dei problemi ... 2744
Capacità insufficiente ... 2744
Impossibile aggiungere i nodi al cluster ... 2744
Accesso non autorizzato o negato (kubectl) ... 2746
`il nome host non corrisponde` ... 2747
`getsockopt: nessun percorso verso l'host` ... 2747
`Le istanze non sono riuscite a entrare nel cluster Kubernetes` ... 2748
Codici di errore dei gruppi di nodi gestiti ... 2748
`Non autorizzato per le immagini` ... 2753
Il nodo è in stato `NotReady` ... 2753
Raccoglitore di log EKS ... 2754
Rete runtime container non pronta ... 2755
Timeout dell'handshake TLS ... 2756
InvalidClientTokenId ... 2757
Prima di poter aggiornare il piano di controllo, i gruppi di nodi devono corrispondere alla versione di Kubernetes ... 2757
Quando si avviano molti nodi, si verificano errori Too Many Requests ... 2758
Risposta agli errori HTTP 401 Unauthorized (Autorizzazione negata) per le richieste del server API Kubernetes ... 2758
La versione della piattaforma Amazon EKS è più avanti di due versioni rispetto all'attuale versione della piattaforma ... 2759
Domande frequenti sull'integrità del cluster e codici di errore con percorsi di risoluzione ... 2762
Progetti relativi ad Amazon EKS ... 2769

Supporto per il software distribuito su EKS ..... 2769
Strumenti di gestione ..... 2770
eksctl ..... 2770
AWS Controller per Kubernetes ..... 2770
kro (Kube Resource Orchestrator) ..... 2771
Argo CD ..... 2771
CD Flux ..... 2771
CDK per Kubernetes ..... 2771
Rete ..... 2772
Plug-in CNI di Amazon VPC per Kubernetes ..... 2772
AWS Load Balancer Controller per Kubernetes ..... 2772
ExternalDNS ..... 2772
Machine learning ..... 2772
Kubeflow ..... 2773
Auto Scaling ..... 2773
Cluster Autoscaler ..... 2773
Karpenter ..... 2773
Escalator ..... 2773
Monitoraggio ..... 2774
Prometheus ..... 2774
Integrazione continua / distribuzione continua ..... 2774
Jenkins X ..... 2774
Nuove caratteristiche e roadmap ..... 2775
Cronologia dei documenti ..... 2776
Contributi ..... 2847
Modificare una singola pagina ..... 2847
Prerequisiti ..... 50
Procedura ..... 103
Panoramica della richiesta di pull ..... 2849
Modifica i file con GitHub ..... 2850
Prerequisiti ..... 50
Procedura ..... 103
Visualizzazione di un feedback sullo stile ..... 2852
Installa Java ..... 2852
Installa l'estensione VS Code Vale ..... 2853
Sincronizzazione di Vale ..... 2853

Visualizza il feedback di stile in VS Code ..... 2853
Creazione di una pagina ..... 2853
Creazione di una pagina ..... 2854
Aggiungi pagina alla navigazione ..... 2854
Inserire link ..... 2854
Collegamento a una pagina o sezione della Guida per l'utente EKS ..... 2855
Collegamento a un'altra guida nei AWS Documenti ..... 2855
Collegamento a una pagina Web esterna ..... 2855
Creare con Amazon Q ..... 2856
Installare Amazon Q con VS Code ..... 2856
Accedi ad Amazon Q ..... 2857
Utilizzare Amazon Q per creare i contenuti ..... 2857
Suggerimenti ..... 2857
Visualizzazione dell'anteprima PR ..... 2858
Visualizzazione dell'anteprima ..... 2858
Limitazioni dell'anteprima ..... 2859
AsciiDoc sintassi ..... 2859
Nuova pagina ..... 2860
Intestazioni ..... 2860
Formattazione del testo ..... 2860
Elenchi ..... 2860
Link ..... 2861
Esempi di codice ..... 2861
Immagini ..... 2861
Tabelle ..... 2862
Callout ..... 2862
Includere altri file ..... 2862
Attributi (simili alle entità) ..... 2863
Procedure ..... 2863

Contribuisci a migliorare questa pagina

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

Per contribuire a questa guida per l'utente, scegli il GitHub  link  Modifica questa pagina su che si trova nel riquadro destro di ogni pagina.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

# Che cosa è Amazon EKS?

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

## Amazon EKS: gestione semplificata di Kubernetes

Amazon Elastic Kubernetes Service (EKS) fornisce un servizio Kubernetes completamente gestito che elimina la complessità del funzionamento dei cluster Kubernetes. Con EKS è possibile:

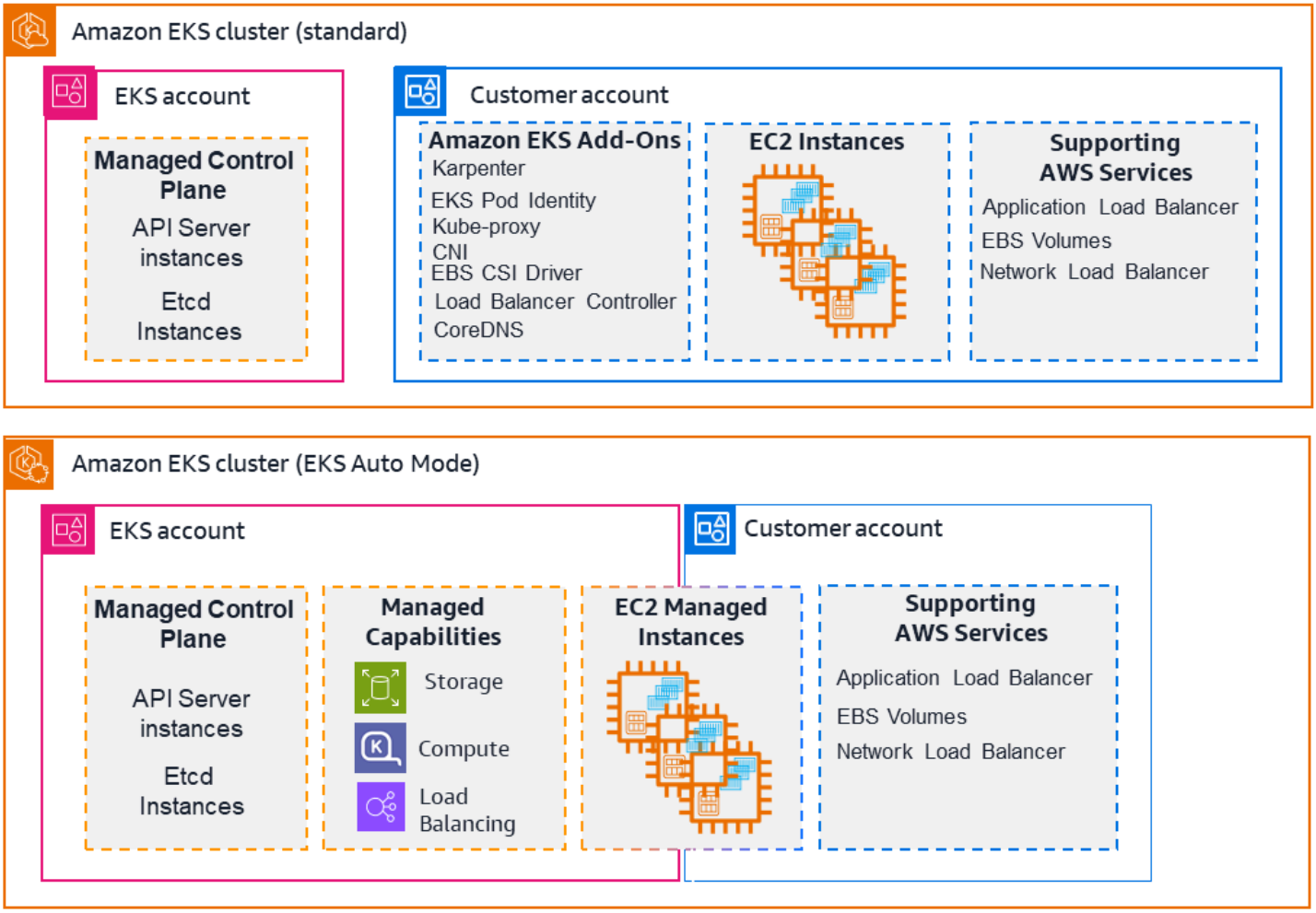
- Implementare le applicazioni più velocemente con meno spese operative
- Scalare senza problemi per soddisfare le mutevoli esigenze dei carichi di lavoro
- Migliora la sicurezza attraverso AWS l'integrazione e gli aggiornamenti automatici
- Scegliere tra la modalità automatica EKS standard o completamente automatizzata

Amazon Elastic Kubernetes Service (Amazon EKS) è la piattaforma principale per l'esecuzione di cluster Kubernetes, sia nel cloud Amazon Web Services () che nei tuoi data center (EKS AWS Anywhere e Amazon EKS Hybrid Nodes).

Amazon EKS semplifica la creazione, la protezione e la manutenzione dei cluster Kubernetes. Fornire risorse sufficienti per soddisfare i picchi di domanda può essere più conveniente rispetto alla manutenzione dei propri data center. Due degli approcci principali all'utilizzo di Amazon EKS sono i seguenti:

- Standard EKS: AWS gestisce il piano di controllo Kubernetes quando crei un cluster con EKS. I componenti che gestiscono i nodi, pianificano i carichi di lavoro, si integrano con il AWS cloud e archiviano e ridimensionano le informazioni del piano di controllo per mantenere i cluster attivi e funzionanti, vengono gestiti automaticamente.
- Modalità automatica EKS: utilizzando la funzione Modalità automatica EKS, EKS estende il proprio controllo anche alla gestione dei nodi (piano dati Kubernetes). Semplifica la gestione di Kubernetes effettuando automaticamente il provisioning dell'infrastruttura, selezionando le istanze di calcolo ottimali, scalando dinamicamente le risorse, ottimizzando continuamente i costi, applicando patch ai sistemi operativi e integrandosi con i servizi di sicurezza. AWS

Il diagramma seguente illustra come Amazon EKS integra i tuoi cluster Kubernetes con il AWS cloud, a seconda del metodo di creazione del cluster scelto:




Amazon EKS ti aiuta a rimuovere gli attriti e ad accelerare i tempi di produzione, migliorare le prestazioni, la disponibilità e la resilienza e migliorare la sicurezza del sistema. Per ulteriori informazioni, consulta Amazon Elastic Kubernetes Service.

## Creazione e scalabilità con Kubernetes: Amazon EKS Capabilities

Amazon EKS non solo ti aiuta a creare e gestire cluster, ma ti aiuta a creare e scalare sistemi applicativi con Kubernetes. Amazon EKS Capabilities sono servizi cluster completamente gestiti che estendono le funzionalità del cluster con Kubernetes-native strumenti a mani libere, tra cui:

- Argo CD: Argo CD fornisce una distribuzione dichiarativa e GitOps-based continua per i carichi di lavoro, le risorse e l'infrastruttura cloud. AWS

- AWS Controller for Kubernetes (ACK): ACK consente la Kubernetes-native creazione e la gestione del ciclo di vita delle risorse, unificando l'orchestrazione dei carichi di lavoro e i flussi di lavoro. AWS Infrastructure-as-code
- kro (Kube Resource Orchestrator): kro estende le funzionalità native di Kubernetes per semplificare la creazione, l'orchestrazione e la composizione di risorse personalizzate, offrendoti gli strumenti per creare i tuoi elementi costitutivi cloud personalizzati.

EKS Capabilities sono risorse cloud che riducono al minimo l'onere operativo di installazione, manutenzione e scalabilità di questi componenti fondamentali della piattaforma nei cluster, consentendoti di concentrarti sulla creazione di software anziché sulle operazioni della piattaforma cluster.

Per ulteriori informazioni, consulta Funzionalità EKS.

# Funzionalità di Amazon EKS

Amazon EKS offre le seguenti funzionalità di alto livello:

Interfacce di gestione

EKS offre diverse interfacce per il provisioning, la gestione e la manutenzione dei cluster, tra cui Console di gestione AWS Amazon EKS API/SDKs, CDK, CLI, eksctl AWS CLI e Terraform. AWS CloudFormation Per ulteriori informazioni, consultare Nozioni di base e Configurazione dei cluster.

Accedere agli strumenti di controllo

EKS si affida alle funzionalità di Kubernetes e AWS Identity and Access Management (AWS IAM) per gestire l'accesso da utenti e carichi di lavoro. Per ulteriori informazioni, consultare the section called “Accesso API Kubernetes” e the section called “Accesso al carico di lavoro a AWS ”.

Risorse di calcolo

Per quanto riguarda le risorse di calcolo, EKS consente l'intera gamma di tipi di istanze Amazon EC2 AWS e innovazioni come Nitro e Graviton con Amazon EKS per ottimizzare l'elaborazione per i tuoi carichi di lavoro. Per ulteriori informazioni, consulta Gestione del calcolo.

Storage

Modalità automatica EKS crea automaticamente classi di archiviazione utilizzando volumi EBS, Utilizzando i driver CSI (Container Storage Interface), puoi anche utilizzare Amazon S3, Amazon

S3 Files, Amazon EFS, Amazon FSx e Amazon File Cache per le tue esigenze di storage delle applicazioni. Per ulteriori informazioni, consulta Archiviazione di dati per app.

Sicurezza

Il modello di responsabilità condivisa è impiegato in relazione alla Sicurezza in Amazon EKS. Per ulteriori informazioni, consulta Best practice di sicurezza, Sicurezza dell'infrastruttura e Sicurezza Kubernetes.

Strumenti di monitoraggio

Utilizza la dashboard di osservabilità per monitorare i cluster Amazon EKS. Gli strumenti di monitoraggio includono Prometheus CloudWatche ADOT Operator. CloudTrail Per ulteriori informazioni su dashboard, server di metriche e altri strumenti, consulta Costi del cluster EKS e Server delle metriche Kubernetes.

Funzionalità del cluster

EKS offre funzionalità di cluster gestite per l'implementazione continua, la gestione delle risorse cloud e la composizione delle risorse basate su innovazioni open source. EKS installa le API Kubernetes nei cluster, ma i controller e gli altri componenti vengono eseguiti in EKS e sono completamente gestiti, fornendo patch, scalabilità e monitoraggio automatizzati. Per ulteriori informazioni, consulta Funzionalità EKS.

Compatibilità e supporto Kubernetes

Amazon EKS è certificato Kubernetes-conformant, quindi puoi distribuire Kubernetes-compatible applicazioni senza refactoring e utilizzare gli strumenti e i plug-in della community Kubernetes. EKS offre supporto standard e supporto esteso per Kubernetes. Per ulteriori informazioni, consulta the section called "Versioni di Kubernetes".

# Servizi correlati

Servizi da utilizzare con Amazon EKS

Puoi utilizzare altri AWS servizi con i cluster che distribuisci con Amazon EKS:

Amazon EC2

Ottenere capacità di elaborazione scalabile, on-demand, con Amazon EC2.

Amazon EBS

Collegare risorse di archiviazione a blocchi scalabili e ad alte prestazioni con Amazon EBS.

Amazon ECR

Archiviare le immagini dei container in modo sicuro con Amazon ECR.

Amazon CloudWatch

Monitora AWS risorse e applicazioni in tempo reale con Amazon CloudWatch.

Amazon Prometheus

Tenere traccia delle metriche per le applicazioni all'interno di container con Servizio gestito da Amazon per Prometheus.

Elastic Load Balancing

Distribuire il traffico in ingresso tra più destinazioni con Elastic Load Balancing.

Amazon GuardDuty

Rileva le minacce ai cluster EKS con Amazon GuardDuty.

AWS Hub di resilienza

Valutare la resilienza dei cluster EKS con AWS Resilience Hub.

# Prezzi di Amazon EKS

Amazon EKS prevede prezzi per cluster basati sul supporto della versione del cluster Kubernetes, prezzi per modalità automatica Amazon EKS e prezzi per vCPU per Amazon EKS Hybrid Nodes.

Quando usi Amazon EKS, paghi separatamente per le AWS risorse utilizzate per eseguire le tue applicazioni sui nodi di lavoro Kubernetes. Per esempio, se utilizzi nodi di lavoro Kubernetes come istanze Amazon EC2 con volumi Amazon EBS e indirizzi IPv4 pubblici, ti verrà addebitata la capacità dell'istanza tramite Amazon EC2, la capacità di volume tramite Amazon EBS e l'indirizzo IPv4 tramite Amazon VPC.

La comunicazione tra il piano di controllo di Amazon EKS e i nodi di lavoro nel cluster utilizza interfacce di rete gestite dai richiedenti nel tuo VPC. Ti vengono addebitate le tariffe di trasferimento AWS dati standard per il traffico sul lato cliente di questa connessione, in particolare, in ingresso ai nodi di lavoro dal piano di controllo e in uscita dai nodi di lavoro al piano di controllo. Amazon EKS assorbe i costi di trasferimento dei dati sul lato del piano di controllo della connessione. Per ulteriori informazioni sui prezzi di trasferimento dei dati, consulta Prezzi per il trasferimento dei dati.

Visita le rispettive pagine dei prezzi dei AWS servizi che utilizzi con le tue applicazioni Kubernetes per informazioni dettagliate sui prezzi.

- Per i prezzi di cluster Amazon EKS, Amazon EKS Auto Mode, Amazon EKS Capabilities e Amazon EKS Hybrid Nodes, consulta i prezzi di Amazon EKS.
- Per i prezzi di Amazon EC2, consulta i prezzi di Amazon On-Demand EC2 e i prezzi Spot di Amazon EC2.
- Per i prezzi di AWS Fargate, consulta i prezzi di Fargate AWS .
- Puoi utilizzare i tuoi savings plans per le risorse di calcolo utilizzate nei cluster Amazon EKS. Per ulteriori informazioni, consulta Prezzi con Savings Plans.

# Casi d'uso comuni in Amazon EKS

Amazon EKS offre solidi servizi Kubernetes gestiti su AWS, progettati per ottimizzare le applicazioni containerizzate. Di seguito sono riportati alcuni dei casi d'uso più comuni di Amazon EKS per aiutarti a sfruttarne i punti di forza per esigenze specifiche.

Implementazione di applicazioni ad alta disponibilità

Utilizzando Elastic Load Balancing, puoi assicurarti che le applicazioni siano altamente disponibili in più zone di disponibilità.

Creazione di architetture di microservizi

Utilizzare le funzionalità di rilevamento servizi di Kubernetes con AWS Cloud Map o Amazon VPC Lattice per creare sistemi resilienti.

Automazione dei processi di rilascio del software

Gestisci le pipeline di integrazione continua e distribuzione continua (CI/CD) che semplificano il processo di creazione, test e implementazione automatizzati delle applicazioni. Per una distribuzione continua dichiarativa, consulta Continuous Deployment with Argo CD.

Esecuzione di applicazioni serverless

Utilizzare AWS Fargate con Amazon EKS per eseguire applicazioni serverless. In questo modo, potrai concentrarti esclusivamente sullo sviluppo delle applicazioni mentre Amazon EKS e Fargate gestiscono l'infrastruttura sottostante.

Esecuzione di carichi di lavoro di machine learning

Amazon EKS è compatibile con i framework di machine learning più diffusi, come TensorFlow, MXNet e PyTorch. Con il supporto GPU, è possibile gestire efficacemente anche attività di machine learning complesse.

Implementazione coerente in locale e nel cloud

Per semplificare l'esecuzione di Kubernetes in ambienti on-premises, è possibile utilizzare gli stessi cluster, funzionalità e strumenti di Amazon EKS per eseguire nodi autogestiti su AWS Outposts o utilizzare Amazon EKS Hybrid Nodes con la propria infrastruttura. Per ambienti autonomi e isolati, è possibile utilizzare Amazon EKS Anywhere per automatizzare la gestione del ciclo di vita dei cluster di Kubernetes sulla propria infrastruttura.

Esecuzione di carichi di lavoro di elaborazione in batch e big data a costi contenuti

Utilizzare le istanze spot per eseguire carichi di lavoro di elaborazione in batch e big data, come Apache Hadoop e Spark, a un costo ridotto. In questo modo puoi sfruttare la EC2 capacità inutilizzata di Amazon a prezzi scontati.

Gestione delle AWS risorse da Kubernetes

Usa AWS Controllers for Kubernetes (ACK) per creare e gestire AWS risorse direttamente dal tuo cluster Kubernetes utilizzando Kubernetes nativo. APIs

Creazione di astrazioni ingegneristiche della piattaforma

Crea Kubernetes personalizzati APIs che compongono più risorse in astrazioni di livello superiore usando kro (Kube Resource Orchestrator).

Proteggere le applicazioni e garantire la conformità

Implementa pratiche di sicurezza solide e mantieni la conformità con Amazon EKS, che si integra con servizi di AWS sicurezza come AWS Identity and Access Management (IAM), Amazon Virtual Private Cloud (Amazon VPC) AWS e Key Management Service AWS (KMS). Ciò garantisce la privacy e la protezione dei dati secondo gli standard del settore.

# Architettura di Amazon EKS

Amazon EKS segue l'architettura generale dei cluster di Kubernetes. Per ulteriori informazioni, consultare la pagina Kubernetes Components nella documentazione Kubernetes. Le seguenti sezioni riassumono alcuni dettagli aggiuntivi sull'architettura di Amazon EKS.

## Piano di controllo

Amazon EKS garantisce che ogni cluster abbia un proprio piano di controllo Kubernetes specifico. Questo design mantiene l'infrastruttura di ciascun cluster separata, senza sovrapposizioni tra cluster o account. AWS La configurazione include:

Componenti distribuiti

Il piano di controllo posiziona almeno due istanze del server API e tre istanze etcd in tre zone di AWS disponibilità all'interno di una regione. AWS

Prestazioni ottimali

Amazon EKS monitora e regola attivamente le istanze del piano di controllo per mantenere le massime prestazioni.

Resilienza

Se un'istanza del piano di controllo fallisce, Amazon EKS la sostituisce rapidamente, utilizzando una zona di disponibilità diversa, se necessario.

Tempo di attività costante

Eseguendo i cluster su più zone di disponibilità, si consegue l'affidabilità dell'Accordo sul livello di servizio (SLA) sulla disponibilità degli endpoint dei server API.

Amazon EKS utilizza Amazon Virtual Private Cloud (Amazon VPC) per limitare il traffico tra i componenti del piano di controllo all'interno di un singolo cluster. I componenti del cluster non possono visualizzare o ricevere comunicazioni da altri cluster o AWS account, tranne quando autorizzati dalle politiche di controllo degli accessi basato sui ruoli (RBAC) di Kubernetes.

## Calcolo

Oltre al piano di controllo, un cluster Amazon EKS dispone di un set di macchine di lavoro chiamate nodi. La selezione del tipo di nodo del cluster Amazon EKS appropriato è fondamentale per soddisfare i propri requisiti specifici e ottimizzare l'utilizzo delle risorse. Amazon EKS offre i seguenti tipi di nodi primari:

Modalità automatica EKS

EKS Auto Mode estende la AWS gestione oltre il piano di controllo per includere il piano dati, automatizzando la gestione dell'infrastruttura del cluster. Integra le funzionalità di base

di Kubernetes come componenti integrati, tra cui scalabilità automatica del calcolo, rete, bilanciamento del carico, DNS, archiviazione e supporto della GPU. La modalità automatica EKS gestisce i nodi in modo dinamico in base alle richieste del carico di lavoro, utilizzando AMI immutabili con funzionalità di sicurezza avanzate. Automatizza aggiornamenti e upgrade rispettando i budget per le interruzioni dei pod e include componenti gestiti che altrimenti richiederebbero la gestione di componenti aggiuntivi. Questa opzione è ideale per gli utenti che desiderano sfruttare le AWS competenze per le operazioni quotidiane, ridurre al minimo il sovraccarico operativo e concentrarsi sullo sviluppo di applicazioni piuttosto che sulla gestione dell'infrastruttura.

AWS Fargate

Fargate è un motore di elaborazione serverless per container che elimina la necessità di gestire le istanze sottostanti. Fargate consente di specificare le esigenze di risorse dell'applicazione e di effettuare AWS automaticamente il provisioning, la scalabilità e la manutenzione dell'infrastruttura. Questa opzione è ideale per gli utenti che danno priorità alla facilità d'uso e vogliono concentrarsi sullo sviluppo e sull'implementazione delle applicazioni anziché sulla gestione dell'infrastruttura.

Karpenter

Karpenter è un cluster autoscaler Kubernetes flessibile e ad alte prestazioni che aiuta a migliorare la disponibilità delle applicazioni e l'efficienza del cluster. Karpenter avvia risorse di calcolo dimensionate in modo ottimale in risposta alla modifica del carico dell'applicazione. Questa opzione consente di fornire risorse di elaborazione just-in-time che soddisfano i requisiti del carico di lavoro.

Gruppi di nodi gestiti

I gruppi di nodi gestiti sono una combinazione di automazione e personalizzazione per la gestione di una raccolta di istanze Amazon EC2 all'interno di un cluster Amazon EKS. AWS si occupa di attività come l'applicazione di patch, l'aggiornamento e il ridimensionamento dei nodi, semplificando gli aspetti operativi. Parallelamente, sono supportati gli argomenti `kubelet` personalizzati, che offrono la possibilità di creare policy avanzate per la gestione della CPU e della memoria. Inoltre, migliorano la sicurezza tramite i ruoli AWS Identity and Access Management (IAM) per gli account di servizio, riducendo al contempo la necessità di autorizzazioni separate per cluster.

Self-managed nodi

Self-managed i nodi offrono il controllo completo sulle istanze Amazon EC2 all'interno di un cluster Amazon EKS. L'utente è responsabile della gestione, del dimensionamento e della

manutenzione dei nodi, disponendo del controllo totale sull'infrastruttura sottostante. Questa opzione è consigliata per gli utenti che necessitano di controllo e personalizzazione granulari dei propri nodi e sono pronti a investire tempo nella gestione e nella manutenzione della propria infrastruttura.

Amazon EKS Hybrid Nodes

Con Amazon EKS Hybrid Nodes è possibile utilizzare l'infrastruttura on-premises ed edge come nodi nei cluster Amazon EKS. Amazon EKS Hybrid Nodes unifica la gestione di Kubernetes tra gli ambienti e affida la gestione del piano di controllo Kubernetes alle applicazioni locali e periferiche. AWS

## Funzionalità EKS

Amazon EKS offre funzionalità di cluster completamente gestite, installando e gestendo le API Kubernetes (con Kubernetes Custom Resource Definitions) nel cluster e gestendo controller e altri componenti in AWS un'infrastruttura di proprietà, separata dal cluster. EKS fornisce patch, scalabilità e monitoraggio automatizzati di queste funzionalità, gestendone completamente il ciclo di vita per ridurre il carico di lavoro dei servizi in cluster per l'orchestrazione dei carichi di lavoro, la gestione delle risorse e altro ancora. AWS

EKS offre i seguenti tipi di funzionalità:

AWS Controller per Kubernetes (ACK)

AWS Controllers for Kubernetes (ACK) ti consente di gestire le AWS risorse utilizzando le API Kubernetes, consentendoti di definire bucket S3, database RDS, ruoli IAM e altre risorse come risorse personalizzate Kubernetes. AWS Puoi gestire AWS le risorse insieme ai carichi di lavoro Kubernetes utilizzando gli stessi strumenti e flussi di lavoro, con supporto per oltre 50 AWS servizi tra cui S3, RDS, DynamoDB e Lambda.

CD Argo

Argo CD implementa la distribuzione GitOps-based continua per i carichi di lavoro delle applicazioni, AWS le risorse e la configurazione del cluster, utilizzando i repository Git come fonte di verità. Argo CD sincronizza automaticamente i cluster con i repository Git e rileva eventuali deviazioni, riconciliandosi continuamente per garantire che le applicazioni e le risorse distribuite corrispondano allo stato desiderato nel controllo delle versioni. È possibile utilizzare Argo CD per gestire le applicazioni su un determinato cluster o distribuire e gestire applicazioni su più cluster

da una singola risorsa Argo CD, con la distribuzione automatica dai repository Git ogni volta che vengono apportate modifiche.

kro (Kube Resource Orchestrator)

kro (Kube Resource Orchestrator) ti consente di creare API Kubernetes personalizzate che compongono più risorse in astrazioni di livello superiore, consentendo ai team della piattaforma di definire modelli riutilizzabili per combinazioni di risorse comuni. Ciò consente ai team della piattaforma di fornire funzionalità self-service con protezioni appropriate, consentendo agli sviluppatori di fornire infrastrutture complesse utilizzando API semplici e create appositamente, mantenendo al contempo gli standard organizzativi e le migliori pratiche.

# Concetti di Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) è un servizio AWS gestito basato sul progetto Kubernetes open source. Sebbene ci siano cose che devi sapere su come il servizio Amazon EKS si integra con il AWS cloud (in particolare quando crei per la prima volta un cluster Amazon EKS), una volta attivo e funzionante, puoi utilizzare il cluster Amazon EKS più o meno allo stesso modo di qualsiasi altro cluster Kubernetes. Quindi, per iniziare a gestire i cluster Kubernetes e distribuire carichi di lavoro, è necessaria almeno una conoscenza di base dei concetti di Kubernetes.

Questa pagina divide i concetti di Kubernetes in tre sezioni: the section called "Perché Kubernetes?", the section called "Cluster" e the section called "Carichi di lavoro". La prima sezione descrive il valore dell'esecuzione di un servizio Kubernetes, in particolare come servizio gestito quale Amazon EKS. La sezione Carichi di lavoro illustra come le applicazioni Kubernetes vengono create, archiviate, eseguite e gestite. La sezione Cluster illustra i diversi componenti che formano i cluster Kubernetes e quali sono le tue responsabilità per la creazione e la manutenzione di questi cluster.

Argomenti


- Perché Kubernetes?
- Cluster
- Carichi di lavoro
- Fasi successive


Durante la lettura di questo contenuto, i link ti condurranno a ulteriori descrizioni dei concetti di Kubernetes nella documentazione di Amazon EKS e Kubernetes, nel caso in cui desideri

approfondire uno degli argomenti trattati qui. Per dettagli su come Amazon EKS implementa il piano di controllo e le funzionalità di calcolo di Kubernetes, consulta the section called "Architecture".

## Perché Kubernetes?

Kubernetes è stato progettato per migliorare la disponibilità e la scalabilità durante l'esecuzione di applicazioni containerizzate mission-critical e di qualità produttiva. Anziché limitarsi a eseguire Kubernetes su una singola macchina (sebbene ciò sia possibile), Kubernetes raggiunge questi obiettivi consentendoti di eseguire applicazioni su set di computer che possono espandersi o contrarsi per soddisfare la domanda. Kubernetes include funzionalità che rendono più semplice:

- Distribuire applicazioni su più macchine (utilizzando container distribuiti nei pod)
- Monitorare lo stato dei container e riavviare quelli che hanno subito un errore
- Aumentare e ridurre i container in base al carico
- Aggiornare i container con le nuove versioni
- Allocare le risorse tra i container
- Bilanciare il traffico tra le macchine

L'automazione di questi tipi di attività complesse da parte di Kubernetes consente agli sviluppatori di applicazioni di concentrarsi sulla creazione e sul miglioramento dei carichi di lavoro delle applicazioni, anziché preoccuparsi dell'infrastruttura. In genere, lo sviluppatore crea file di configurazione, formattati come file YAML, che descrivono lo stato desiderato dell'applicazione. Ciò potrebbe includere i contenitori da eseguire, i limiti delle risorse, il numero di repliche Pod, l' CPU/memory allocazione, le regole di affinità e altro ancora.

### Attributi di Kubernetes

Per raggiungere gli obiettivi, Kubernetes ha i seguenti attributi:

- Containerizzato: Kubernetes è uno strumento di orchestrazione dei container. Per utilizzare Kubernetes, devi prima avere le tue applicazioni containerizzate. A seconda del tipo di applicazione, potrebbe trattarsi di un insieme di microservizi, di processi in batch o in altre forme. Quindi, le tue applicazioni possono trarre vantaggio da un flusso di lavoro di Kubernetes che comprende un enorme ecosistema di strumenti, in cui i container possono essere archiviati come immagini in un registro di container, distribuiti in un cluster Kubernetes ed eseguiti su un nodo disponibile. Puoi creare e testare singoli container sul tuo computer locale con Docker o un altro runtime di container prima di distribuirli nel cluster Kubernetes.

- Scalabile: se la domanda per le tue applicazioni supera la capacità delle istanze in esecuzione delle stesse, Kubernetes è in grado di aumentare verticalmente. Se necessario, Kubernetes è in grado di stabilire se le applicazioni richiedono più CPU o memoria e rispondere o espandendo automaticamente la capacità disponibile o utilizzando una maggiore capacità esistente. La scalabilità può essere eseguita a livello di pod, se è disponibile una quantità di calcolo sufficiente per eseguire solo più istanze dell'applicazione (scalabilità automatica del pod in orizzontale), o a livello di nodo, se è necessario attivare più nodi per gestire l'aumento della capacità (Cluster Autoscaler o Karpenter). Poiché la capacità non è più necessaria, questi servizi possono eliminare i pod non necessari e chiudere i nodi non necessari.
- Disponibile: se un'applicazione o un nodo non sono integri o non sono disponibili, Kubernetes può spostare i carichi di lavoro in esecuzione su un altro nodo disponibile. Puoi forzare il problema semplicemente eliminando un'istanza in esecuzione di un carico di lavoro o di un nodo che esegue i tuoi carichi di lavoro. In conclusione, i carichi di lavoro possono essere trasferiti in altre posizioni se non possono più essere eseguiti dove si trovano.
- Dichiarativo: Kubernetes utilizza la riconciliazione attiva per verificare costantemente che lo stato dichiarato per il cluster corrisponda allo stato effettivo. Applicando oggetti Kubernetes a un cluster, in genere tramite file di YAML-formatted configurazione, puoi, ad esempio, chiedere di avviare i carichi di lavoro che desideri eseguire sul tuo cluster. Successivamente, puoi modificare le configurazioni per eseguire operazioni quali utilizzare una versione successiva di un container o allocare più memoria. Kubernetes farà ciò che deve fare per stabilire lo stato desiderato. Ciò può includere l'attivazione o la disattivazione dei nodi, l'arresto e il riavvio dei carichi di lavoro o l'estrazione di container aggiornati.
- Componibile: poiché un'applicazione è in genere composta da più componenti, è necessario essere in grado di gestire insieme un insieme di questi componenti (spesso rappresentati da più container). Sebbene Docker Compose offra un modo per farlo direttamente con Docker, il comando Kompose di Kubernetes può aiutarti a farlo con Kubernetes. Consultare Translate a Docker Compose File to Kubernetes Resources per un esempio di come eseguire questa operazione.
- Estensibile: a differenza del software proprietario, il progetto open source Kubernetes è progettato per consentirti di estendere Kubernetes in qualsiasi modo desideri per soddisfare le tue esigenze. Le API e i file di configurazione sono suscettibili di modifiche dirette. Third-parties sono incoraggiati a scrivere i propri controller, per estendere sia l'infrastruttura che le funzionalità di Kubernetes per gli utenti finali. I webhook consentono di configurare le regole del cluster per applicare le policy e adattarsi alle condizioni mutevoli. Per altre idee su come estendere i cluster Kubernetes, consultare Extending Kubernetes.

- Portatile: molte organizzazioni hanno standardizzato le proprie operazioni su Kubernetes perché consente loro di gestire tutte le esigenze applicative nello stesso modo. Gli sviluppatori possono utilizzare le stesse pipeline per creare e archiviare applicazioni containerizzate. Tali applicazioni possono quindi essere distribuite su cluster Kubernetes in esecuzione on-premise, nei cloud, sui terminali dei punti vendita nei ristoranti o su dispositivi IoT sparsi nei siti remoti di un'azienda. La sua natura open source consente alle persone di sviluppare queste speciali distribuzioni Kubernetes, insieme agli strumenti necessari per gestirle.

## Gestione di Kubernetes

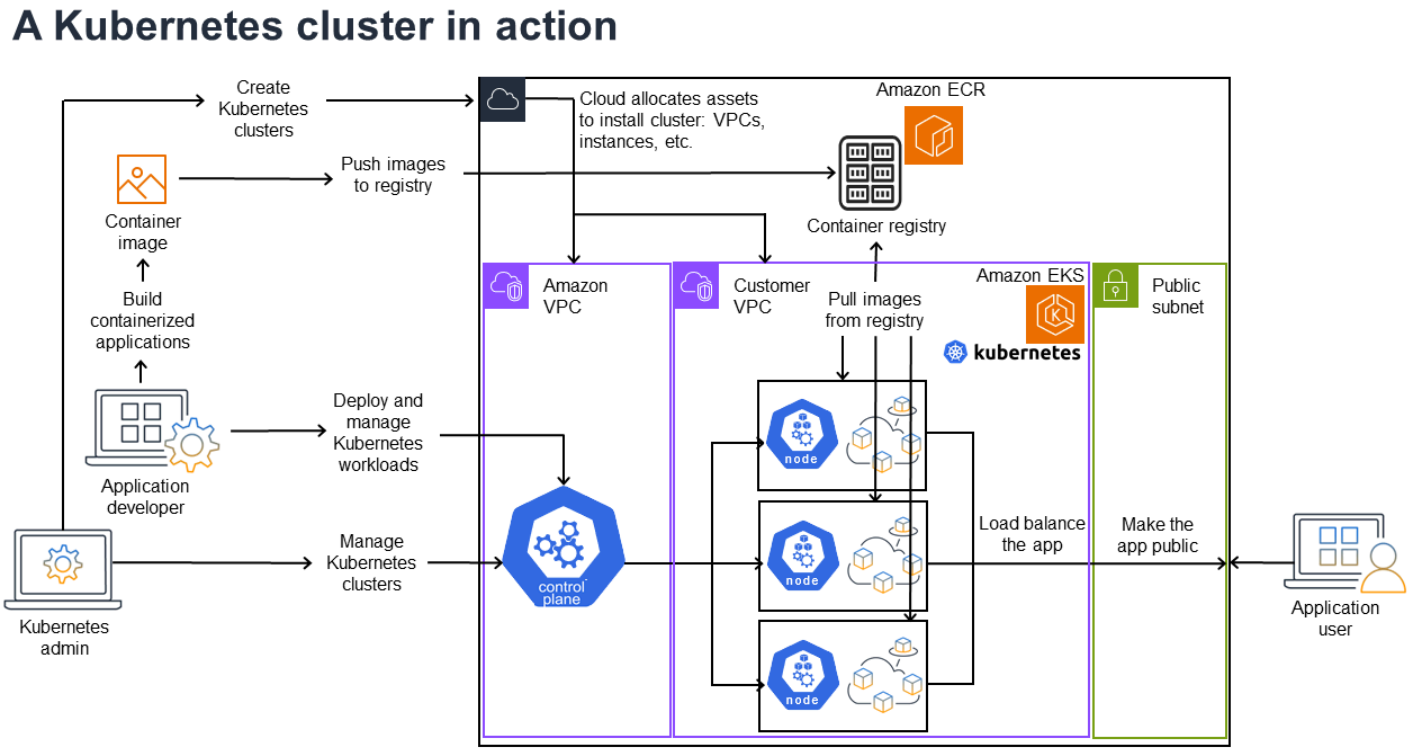
Il codice sorgente di Kubernetes è disponibile gratuitamente, quindi è possibile installare e gestire Kubernetes da soli con le proprie apparecchiature. Tuttavia, la gestione automatica di Kubernetes richiede una profonda esperienza operativa e richiede tempo e impegno per la manutenzione. Per questi motivi, la maggior parte delle persone che implementano carichi di lavoro di produzione sceglie un provider cloud (come Amazon EKS) o un provider on-premises (come Amazon EKS Anywhere) con la propria distribuzione Kubernetes testata e il supporto di esperti Kubernetes. Ciò consente di alleggerire gran parte del carico di lavoro indifferenziato necessario per la manutenzione dei cluster, tra cui:

- Hardware: se non disponi di hardware disponibile per eseguire Kubernetes in base alle tue esigenze, un provider di servizi cloud come AWS Amazon EKS può farti risparmiare sui costi iniziali. Con Amazon EKS, ciò significa che puoi utilizzare le migliori risorse cloud offerte AWS, tra cui le istanze di calcolo (Amazon Elastic Compute Cloud), il tuo ambiente privato (Amazon VPC), la gestione centralizzata delle identità e delle autorizzazioni (IAM) e lo storage (Amazon EBS). AWS gestisce i computer, le reti, i data center e tutti gli altri componenti fisici necessari per eseguire Kubernetes. Allo stesso modo, non è necessario pianificare il data center per gestire la capacità massima nei giorni con una maggiore richiesta. Per Amazon EKS Anywhere o altri cluster Kubernetes locali, sei responsabile della gestione dell'infrastruttura utilizzata nelle tue distribuzioni Kubernetes, ma puoi comunque contare su di te per mantenere Kubernetes aggiornato. AWS
- Gestione del piano di controllo: Amazon EKS gestisce la sicurezza e la disponibilità del piano di controllo Kubernetes AWS ospitato, responsabile della pianificazione dei contenitori, della gestione della disponibilità delle applicazioni e di altre attività chiave, così puoi concentrarti sui carichi di lavoro delle applicazioni. Se il cluster si interrompe, AWS dovrebbe avere i mezzi per ripristinare lo stato di esecuzione del cluster. Per Amazon EKS Anywhere, il piano di controllo verrebbe gestito dall'utente stesso.

- Upgrade testati: quando aggiorni i tuoi cluster, puoi fare affidamento su Amazon EKS o Amazon EKS Anywhere per fornire versioni testate delle distribuzioni Kubernetes.
- Add-ons— Esistono centinaia di progetti creati per estendere e funzionare con Kubernetes che puoi aggiungere all'infrastruttura del tuo cluster o utilizzare per facilitare l'esecuzione dei tuoi carichi di lavoro. Invece di creare e gestire personalmente questi componenti aggiuntivi, è possibile AWS utilizzarli con the section called "Componenti aggiuntivi Amazon EKS" i cluster. Amazon EKS Anywhere fornisce pacchetti curati che includono build di molti progetti open source popolari. Quindi non è necessario creare il software o gestire patch di sicurezza, correzioni di bug o aggiornamenti critici per proprio conto. Allo stesso modo, se le impostazioni predefinite soddisfano le tue esigenze, è normale che sia necessaria una configurazione minima di tali componenti aggiuntivi. Consultare the section called "Estendere i cluster" per i dettagli sull'estensione del cluster con componenti aggiuntivi.

## Kubernetes in azione

Il diagramma seguente mostra le attività chiave che potresti svolgere in qualità di amministratore o sviluppatore di applicazioni Kubernetes per creare e utilizzare un cluster Kubernetes. Nel processo, illustra come i componenti di Kubernetes interagiscono tra loro, utilizzando il AWS cloud come esempio del provider cloud sottostante.



Un amministratore Kubernetes crea il cluster Kubernetes utilizzando uno strumento specifico per il tipo di provider su cui verrà creato il cluster. Questo esempio utilizza il AWS cloud come provider, che offre il servizio Kubernetes gestito chiamato Amazon EKS. Il servizio gestito alloca automaticamente le risorse necessarie per creare il cluster, inclusa la creazione di due nuovi cloud privati virtuali (Amazon VPC) per il cluster, la configurazione della rete e la mappatura delle autorizzazioni Kubernetes direttamente nei nuovi VPC per la gestione delle risorse cloud. Il servizio gestito rileva inoltre che i servizi del piano di controllo hanno posizioni in cui eseguire e alloca zero o più istanze Amazon EC2 come nodi Kubernetes per l'esecuzione dei carichi di lavoro. AWS gestisce autonomamente un Amazon VPC per il piano di controllo, mentre l'altro Amazon VPC contiene i nodi del cliente che eseguono i carichi di lavoro.

Molte delle attività future dell'amministratore di Kubernetes vengono eseguite utilizzando strumenti Kubernetes come `kubectl`. Questo strumento invia le richieste di servizi direttamente al piano di controllo del cluster. I modi in cui le query e le modifiche vengono apportate al cluster sono quindi molto simili a quelli in cui verrebbero eseguite in qualsiasi cluster Kubernetes.

Uno sviluppatore di applicazioni che desidera distribuire carichi di lavoro in questo cluster può eseguire diverse attività. Lo sviluppatore deve creare l'applicazione in una o più immagini del contenitore, quindi inviarle a un registro di container accessibile al cluster Kubernetes. AWS offre Amazon Elastic Container Registry (Amazon ECR) a tale scopo.

Per eseguire l'applicazione, lo sviluppatore può creare file di YAML-formatted configurazione che indicano al cluster come eseguire l'applicazione, inclusi quali contenitori estrarre dal registro e come racchiudere tali contenitori in Pods. Il piano di controllo (scheduler) pianifica i container su uno o più nodi e il runtime del container su ciascun nodo recupera ed esegue effettivamente i container necessari. Lo sviluppatore può anche configurare un Application Load Balancer per bilanciare il traffico verso i container disponibili in esecuzione su ciascun nodo ed esporre l'applicazione al mondo esterno in modo che sia disponibile su una rete pubblica. Fatto ciò, qualcuno che desidera utilizzare l'applicazione può connettersi al suo endpoint per accedervi.

Le sezioni seguenti illustrano i dettagli di ciascuna di queste funzionalità, dal punto di vista dei cluster e dei carichi di lavoro Kubernetes.

## Cluster

Se il tuo compito è avviare e gestire i cluster Kubernetes, devi sapere come vengono creati, migliorati, gestiti ed eliminati. Devi anche sapere quali sono i componenti che formano un cluster e cosa fare per mantenerli.

Gli strumenti per la gestione dei cluster si occupano della sovrapposizione tra i servizi Kubernetes e il provider hardware sottostante. Per questo motivo, l'automazione di queste attività tende a essere eseguita dal provider Kubernetes (come Amazon EKS o Amazon EKS Anywhere) utilizzando strumenti specifici del provider. Ad esempio, per avviare un cluster Amazon EKS si può utilizzare `eksctl create cluster`, mentre per Amazon EKS Anywhere è possibile utilizzare `eksctl anywhere create cluster`. Nota che, sebbene questi comandi creino un cluster Kubernetes, sono specifici del provider e non fanno parte del progetto Kubernetes stesso.

## Strumenti per la creazione e la gestione dei cluster

Il progetto Kubernetes offre strumenti per creare manualmente un cluster Kubernetes. Quindi, se vuoi installare Kubernetes su una singola macchina o eseguire il piano di controllo su una macchina e aggiungere nodi manualmente, puoi utilizzare strumenti CLI come kind, minikube o kubeadm elencati in Install Tools di Kubernetes. Per semplificare e automatizzare l'intero ciclo di vita della creazione e della gestione dei cluster, è molto più semplice utilizzare gli strumenti supportati da un provider Kubernetes affermato, come Amazon EKS o Amazon EKS Anywhere.

Nel AWS cloud, puoi creare cluster Amazon EKS utilizzando strumenti CLI, come eksctl, o strumenti più dichiarativi, come Terraform (vedi Amazon EKS Blueprints for Terraform). https://github.com/aws-ia/terraform-aws-eks-blueprints Puoi creare un cluster da Console di gestione AWS. Consultare Amazon EKS features per un elenco di ciò che si ottiene con Amazon EKS. Le responsabilità di Kubernetes che Amazon EKS si assume per te includono:

- Piano di controllo gestito:AWS assicura che il cluster Amazon EKS sia disponibile e scalabile perché gestisce il piano di controllo per te e lo rende disponibile nelle zone di disponibilità. AWS
- Gestione dei nodi: invece di aggiungere i nodi manualmente, puoi fare in modo che Amazon EKS crei i nodi in automatico a seconda delle necessità, utilizzando Gruppi di nodi gestiti (vedere the section called "Gruppi di nodi gestiti") o Karpenter. I gruppi di nodi gestiti presentano integrazioni con Cluster Autoscaling di Kubernetes. Utilizzando gli strumenti di gestione dei nodi, è possibile trarre vantaggio dai risparmi sui costi, come le istanze spot e il consolidamento e la disponibilità dei nodi, utilizzando le funzionalità di pianificazione per impostare la modalità di distribuzione dei carichi di lavoro e la selezione dei nodi.
- Rete in cluster: utilizzando CloudFormation modelli, `eksctl` configura il networking tra i componenti del piano di controllo e del piano dati (nodo) nel cluster Kubernetes. Inoltre, configura gli endpoint attraverso i quali possono avvenire le comunicazioni interne ed esterne. Per ulteriori informazioni, consulta De-mystifying Cluster Networking Nodes per i nodi di lavoro di Amazon EKS. Le comunicazioni tra Pod in Amazon EKS vengono effettuate utilizzando Amazon EKS Pod

Identities (vedithe section called "Pod Identity"), che fornisce un mezzo per consentire ai Pods di attingere ai metodi AWS cloud di gestione delle credenziali e delle autorizzazioni.

- Add-Ons— Amazon EKS ti evita di dover creare e aggiungere componenti software comunemente usati per supportare i cluster Kubernetes. Ad esempio, quando si crea un cluster Amazon EKS da Console di gestione AWS, vengono aggiunti automaticamente i componenti aggiuntivi Amazon EKS kube-proxy (the section called "kube-proxy"), Amazon VPC CNI per Kubernetes () e CoreDNS (). the section called "CNI di Amazon VPC" the section called "CoreDNS" Consultare the section called "Componenti aggiuntivi Amazon EKS" per ulteriori elementi aggiuntivi, incluso un elenco di quali sono disponibili.

Per eseguire i cluster su computer e reti on-premises, Amazon offre Amazon EKS Anywhere. Invece che il AWS cloud sia il provider, puoi scegliere di eseguire Amazon EKS Anywhere su piattaforme VMware vSphere, bare metal (provider Tinkerbell) CloudStack, Snow o Nutanix utilizzando le tue apparecchiature.

Amazon EKS Anywhere si basa sullo stesso software Amazon EKS Distro utilizzato da Amazon EKS. Tuttavia, Amazon EKS Anywhere si basa su diverse implementazioni dell'interfaccia Kubernetes Cluster API (CAPI) per gestire l'intero ciclo di vita delle macchine in un cluster Amazon EKS Anywhere (come CAPV per vSphere e CAPC per CloudStack). Poiché l'intero cluster è in esecuzione sulle tue apparecchiature, ti assumi la responsabilità aggiuntiva della gestione del piano di controllo e del backup dei relativi dati (consulta `etcd` più avanti in questo documento).

## Componenti del cluster

I componenti del cluster Kubernetes sono suddivisi in due aree principali: piano di controllo e nodi worker. I componenti del piano di controllo gestiscono il cluster e forniscono l'accesso alle relative API. I nodi worker (a volte denominati semplicemente nodi) forniscono i luoghi in cui vengono eseguiti i carichi di lavoro effettivi. I componenti del nodo sono costituiti da servizi eseguiti su ciascun nodo per comunicare con il piano di controllo ed eseguire i container. Il set di nodi worker per il cluster è denominato Piano dati.

### Piano di controllo

Il piano di controllo è costituito da un insieme di servizi che gestiscono il cluster. Questi servizi possono essere eseguiti tutti su un singolo computer o possono essere distribuiti su più computer. Internamente, queste sono denominate Istanze del piano di controllo (CPI). Il modo in cui vengono eseguite le CPI dipende dalla dimensione del cluster e dai requisiti di elevata disponibilità. Con

l'aumento della domanda nel cluster, un servizio del piano di controllo può scalare per fornire più istanze di quel servizio, con il bilanciamento del carico delle richieste tra le istanze.

Le attività eseguite dai componenti del piano di controllo di Kubernetes includono:

- Comunicazione con i componenti del cluster (server API): il server API (kube-apiserver) espone l'API Kubernetes in modo che le richieste al cluster possano essere effettuate sia dall'interno che dall'esterno del cluster. In altre parole, le richieste di aggiunta o modifica degli oggetti di un cluster (pod, servizi, nodi e così via) possono provenire da comandi esterni, come le richieste di esecuzione da parte di `kubectl` di eseguire un pod. Allo stesso modo, è possibile effettuare richieste dal server API ai componenti all'interno del cluster, ad esempio una richiesta al servizio `kubelet` per lo stato di un pod.
- Archivia dati sul cluster (archivio di valori delle chiavi **`etcd`**): il servizio `etcd` svolge il ruolo fondamentale di tenere traccia dello stato attuale del cluster. Se il servizio `etcd` diventasse inaccessibile, non sarebbe possibile aggiornare lo stato del cluster o eseguire query su di esso, anche se i carichi di lavoro continuerebbero a funzionare per un po'. Per questo motivo, i cluster critici in genere dispongono di più istanze di bilanciatore del carico del servizio `etcd` in esecuzione contemporaneamente ed effettuano backup periodici dell'archivio di valori delle chiavi `etcd` in caso di perdita o danneggiamento dei dati. Tieni presente che, in Amazon EKS, tutto questo viene gestito automaticamente per impostazione predefinita. Amazon EKS Anywhere fornisce istruzioni per etcd backup and restore. Consulta etcd Data Model per scoprire come `etcd` gestisce i dati.
- Programma i pod ai nodi (Scheduler): le richieste di avvio o arresto di un pod in Kubernetes vengono indirizzate a Kubernetes Scheduler (kube-scheduler). Poiché un cluster può avere più nodi in grado di eseguire il pod, spetta allo Scheduler scegliere su quale nodo (o nodi, nel caso delle repliche) eseguire il pod. Se la capacità disponibile non è sufficiente per eseguire il pod richiesto su un nodo esistente, la richiesta avrà esito negativo, a meno che non siano state prese altre disposizioni. Tali disposizioni potrebbero includere l'abilitazione di servizi come Gruppi di nodi gestiti (the section called "Gruppi di nodi gestiti") o Karpenter in grado di avviare automaticamente nuovi nodi per gestire i carichi di lavoro.
- Mantieni i componenti nello stato desiderato (Gestore di controller): il Gestore controller di Kubernetes viene eseguito come processo daemon (kube-controller-manager) per monitorare lo stato del cluster e apportare modifiche al cluster per ristabilire gli stati previsti. In particolare, esistono diversi controller che controllano diversi oggetti Kubernetes, tra cui `statefulset-controller`, `endpoint-controller`, `cronjob-controller`, `node-controller` e altri.
- Gestisci le risorse cloud (Gestore di controller cloud): le interazioni tra Kubernetes e il provider cloud che esegue le richieste per le risorse del data center sottostanti vengono gestite dal Gestore

di controller cloud (cloud-controller-manager). I controller gestiti da Gestore di controller cloud possono includere un controller di percorso (per configurare i percorsi di rete cloud), un controller di servizio (per l'utilizzo dei servizi di bilanciatore del carico nel cloud) e un controller del ciclo di vita dei nodi (per mantenere i nodi sincronizzati con Kubernetes per tutto il loro ciclo di vita).

Nodi worker (piano dati)

Per un cluster Kubernetes a nodo singolo, i carichi di lavoro vengono eseguiti sulla stessa macchina del piano di controllo. Tuttavia, una configurazione più standard prevede uno o più sistemi informatici separati (nodi) dedicati all'esecuzione dei carichi di lavoro Kubernetes.

Quando crei per la prima volta un cluster Kubernetes, alcuni strumenti per la creazione di cluster ti consentono di configurare un certo numero di nodi da aggiungere al cluster (identificando i sistemi informatici esistenti o chiedendo al provider di crearne di nuovi). Prima di aggiungere qualsiasi carico di lavoro a tali sistemi, vengono aggiunti servizi a ciascun nodo per implementare queste funzionalità:

- Gestisci ogni nodo (**`kubelet`**): il server API comunica con il servizio kubelet in esecuzione su ciascun nodo per assicurarsi che il nodo sia registrato correttamente e che i pod richiesti dallo Scheduler siano in esecuzione. Il kubelet può leggere i manifest dei pod e configurare volumi di archiviazione o altre funzionalità necessarie ai pod sul sistema locale. Può anche verificare lo stato dei container in esecuzione locale.
- Esegui i container su un nodo (runtime del container): il Runtime del container su ciascun nodo gestisce i container richiesti per ogni pod assegnato al nodo. Ciò significa che può estrarre le immagini del contenitore dal registro appropriato, eseguire il contenitore, fermarlo e rispondere alle domande sul contenitore. Il runtime predefinito del container è containerd. A partire da Kubernetes 1.24, la speciale integrazione di Docker (`dockershim`) che poteva essere utilizzata come runtime del container è stata eliminata da Kubernetes. Sebbene sia ancora possibile utilizzare Docker per testare ed eseguire container sul sistema locale, per utilizzare Docker con Kubernetes ora è necessario installare il motore Docker su ogni nodo per utilizzarlo con Kubernetes.
- Gestisci la rete tra container (**`kube-proxy`**): per supportare la comunicazione tra i pod, Kubernetes utilizza una funzionalità denominata Servizio per configurare le reti pod che tengono traccia degli indirizzi IP e delle porte associate a tali pod. Il servizio kube-proxy viene eseguito su ogni nodo per consentire la comunicazione tra i pod.

### Estendere i cluster

Esistono alcuni servizi che puoi aggiungere a Kubernetes per supportare il cluster, ma non vengono eseguiti nel piano di controllo. Questi servizi spesso vengono eseguiti direttamente sui nodi del namespace kube-system o nel relativo namespace (come spesso accade con fornitori di servizi di terzi). Un esempio comune è il servizio CoreDNS, che fornisce servizi DNS al cluster. Fai riferimento a Discovering builtin services per informazioni su come vedere quali servizi cluster sono in esecuzione in kube-system sul tuo cluster.

Esistono diversi tipi di componenti aggiuntivi che puoi considerare di aggiungere ai tuoi cluster. Per mantenere integri i cluster, puoi aggiungere funzionalità di osservabilità (consultare Monitoraggio dei cluster) che consentono di eseguire operazioni come la registrazione, il controllo e le metriche. Con queste informazioni, puoi risolvere i problemi che si verificano, spesso tramite le stesse interfacce di osservabilità. Esempi di questi tipi di servizi includono Amazon GuardDuty, CloudWatch (vedithe section called "Amazon CloudWatch"), AWS Distro for OpenTelemetry, il plugin Amazon VPC CNI per Kubernetes (vedithe section called "CNI di Amazon VPC") e Grafana Kubernetes Monitoring. https://grafana.com/docs/grafana-cloud/monitor-infrastructure/kubernetes-monitoring/configuration/config-aws-eks/ Per quanto riguarda lo storage (vediArchiviazione di dati per app), i componenti aggiuntivi di Amazon EKS includono Amazon Elastic Block Store CSI Driver (vedithe section called "Amazon EBS"), Amazon Elastic File System CSI Driver (vedi) e diversi componenti aggiuntivi di storage di terze parti come il driver Amazon FSx for ONTAP CSIthe section called "Amazon EFS"). NetApp the section called "Amazon FSx per ONTAP NetApp "

Per un elenco più completo dei componenti aggiuntivi di Amazon EKS, consultare the section called "Componenti aggiuntivi Amazon EKS".

## Carichi di lavoro

Kubernetes definisce un carico di lavoro come "un'applicazione in esecuzione su Kubernetes". Tale applicazione può essere costituita da un set di micro servizi eseguiti come container in pod oppure può essere eseguita come processo batch o un altro tipo di applicazioni. Il compito di Kubernetes è assicurarsi che le richieste effettuate per la configurazione o la distribuzione di tali oggetti vengano eseguite. Chi distribuisce le applicazioni dovrebbe imparare come vengono creati i container, come vengono definiti i pod e quali metodi è possibile utilizzare per distribuirli.

### Container

L'elemento più basilare del carico di lavoro di un'applicazione che distribuisci e gestisci in Kubernetes è un pod . Un pod rappresenta un modo per contenere i componenti di un'applicazione e per definire

le specifiche che descrivono gli attributi del pod. Confrontalo con qualcosa come un pacchetto RPM o Deb, che raggruppa il software per un sistema Linux, ma non funziona come un'entità.

Poiché il pod è l'unità implementabile più piccola, in genere contiene un singolo container. Tuttavia, in un pod possono essere presenti più container nei casi in cui questi siano strettamente collegati. Ad esempio, un container di server Web potrebbe essere impacchettato in un pod con un tipo di container secondario che può fornire la registrazione, il monitoraggio o altri servizi strettamente legati al container del server Web. In questo caso, la presenza nello stesso pod garantisce che per ogni istanza in esecuzione del pod, entrambi i container vengano sempre eseguiti sullo stesso nodo. Allo stesso modo, tutti i container in un pod condividono lo stesso ambiente, con i container in un pod che funzionano come se si trovassero nello stesso host isolato. L'effetto è che i container condividono un unico indirizzo IP che fornisce l'accesso al pod e possono comunicare tra loro come se fossero in esecuzione sul proprio host locale.

Le specifiche del pod (PodSpec) definiscono lo stato desiderato del Pod. Puoi implementare uno o più pod utilizzando le risorse del carico di lavoro per gestire i modelli di pod. Le risorse per il carico di lavoro includono le implementazioni (per gestire più repliche di pod), StatefulSets (per distribuire pod che devono essere unici, come i pod di database) e DaemonSets (in cui un pod deve essere eseguito continuamente su ogni nodo). Ne parleremo più avanti.

Sebbene un pod sia l'unità più piccola implementabile, un container è l'unità più piccola da costruire e gestire.

Costruire container

Il pod è in realtà solo una struttura attorno a uno o più container, dove ognuno di essi contiene il file system, gli eseguibili, i file di configurazione, le librerie e altri componenti per eseguire effettivamente l'applicazione. Poiché i container sono diventati popolari grazie a una società chiamata Docker Inc., alcune persone si riferiscono ai container come container Docker. Tuttavia, da allora l'Open Container Initiative ha definito i runtime, le immagini e i metodi di distribuzione dei container per il settore. A ciò si aggiunge il fatto che i container sono stati creati partendo da molte funzionalità Linux esistenti, per cui spesso le persone si riferiscono ai contenitori come container OCI, container Linux o semplicemente container.

Quando si crea un container, in genere si inizia con un Dockerfile (chiamato letteralmente così). All'interno di quel Dockerfile, si identifica:

- Un'immagine di base: un'immagine container di base è un container che viene in genere creato da una versione minima del file system di un sistema operativo (come Red Hat Enterprise Linux o

Ubuntu) o da un sistema minimale migliorato per fornire software al fine di eseguire tipi specifici di applicazioni (come app nodejs o python).

- Software applicativo: puoi aggiungere il tuo software applicativo al container più o meno nello stesso modo in cui lo aggiungeresti a un sistema Linux. Ad esempio, nel tuo Dockerfile puoi eseguire `npm` e installare un' JavaScript applicazione o `yarn` installare pacchetti RPM. `yum` `dnf` In altre parole, utilizzando un comando RUN in un Dockerfile, è possibile eseguire qualsiasi comando disponibile nel file system dell'immagine di base per installare software o configurare software all'interno dell'immagine del container risultante.
- Istruzioni: il Dockerfile reference descrive le istruzioni che è possibile aggiungere a un Dockerfile durante la configurazione. Queste includono le istruzioni utilizzate per creare ciò che è contenuto nel container stesso (file `ADD` o `COPY` dal sistema locale), identificare i comandi da eseguire quando il container viene eseguito (`CMD` o `ENTRYPOINT`) e connettere il container al sistema su cui viene eseguito (identificando `USER` eseguire come, un `VOLUME` locale da montare o le porte su `EXPOSE`).

Sebbene il comando e il servizio `docker` siano stati tradizionalmente utilizzati per creare container (`docker build`), altri strumenti disponibili per creare immagini di contenitori includono podman e nerdctl. Consultare Creare immagini di container migliori o Panoramica di Docker Build per saperne di più sulla creazione di container.

Archiviazione dei container

Dopo aver creato l'immagine del container, puoi archiviarla in un registro di distribuzione dei container sulla tua workstation o in un registro pubblico dei container. L'esecuzione di un registro di container privato sulla workstation consente di archiviare le immagini dei container in locale, rendendole immediatamente disponibili.

Per archiviare le immagini dei container in modo più pubblico, puoi inviarle a un registro di container pubblico. I registri pubblici dei container forniscono una posizione centrale per l'archiviazione e la distribuzione delle immagini dei container. Esempi di registri di container pubblici includono Amazon Elastic Container Registry, il registro Red Hat Quay e il registro Docker Hub.

Quando esegui carichi di lavoro containerizzati su Amazon Elastic Kubernetes Service (Amazon EKS), consigliamo di estrarre copie delle immagini ufficiali Docker archiviate in Amazon Elastic Container Registry. Amazon ECR archivia queste immagini dal 2021. Puoi cercare le immagini dei container più comuni nella Galleria pubblica di Amazon ECR e, in particolare, per le immagini di Docker Hub, puoi cercare nella Galleria docker di Amazon ECR.

### Esecuzione dei container

Poiché i container sono creati in un formato standard, un container può essere eseguito su qualsiasi macchina in grado di eseguire un runtime di container (come Docker) e il cui contenuto corrisponda all'architettura della macchina locale (come `x86_64` o `arm`). Per testare un container o semplicemente eseguirlo sul desktop locale, puoi utilizzare i comandi `docker run` o `podman run` per avviare un container su host locale. Per Kubernetes, tuttavia, ogni nodo worker dispone di un runtime di container implementato e spetta a Kubernetes richiedere che un nodo esegua un container.

Una volta che un container è stato assegnato per l'esecuzione su un nodo, quest'ultimo verifica se la versione richiesta dell'immagine del container esiste già sul nodo. In caso contrario, Kubernetes indica al runtime del container di estrarlo dal registro di container appropriato, quindi di eseguirlo localmente. Tieni presente che un'immagine del container si riferisce al pacchetto software che viene spostato tra il laptop, il registro dei container e i nodi Kubernetes. Un container si riferisce a un'istanza in esecuzione di quell'immagine.

## Pod

Una volta che i container sono pronti, l'utilizzo dei pod include la sua configurazione, implementazione e accessibilità.

### Configurazione dei pod

Quando definisci un pod, gli assegni una serie di attributi. Questi attributi devono includere almeno il nome del pod e l'immagine del container da eseguire. Tuttavia, ci sono anche molte altre cose che vuoi configurare con le definizioni del tuo Pod (consulta la PodSpec pagina per i dettagli su cosa può essere inserito in un Pod). Ciò include:

- Archiviazione: quando un container in esecuzione viene interrotto ed eliminato, l'archiviazione di dati in quel container scompare, a meno che non si configuri uno spazio di archiviazione più permanente. Kubernetes supporta molti tipi di storage diversi e li riassume sotto il termine Volumi. I tipi di storage includono CephFS, NFS, iSCSI e altri. È anche possibile utilizzare un dispositivo a blocchi locale dal computer locale. Con uno di questi tipi di archiviazione disponibili nel cluster, è possibile montare il volume di archiviazione su un punto di montaggio selezionato nel file system del container. Un volume persistente è quello che continua a esistere dopo l'eliminazione del pod, mentre un volume temporaneo viene eliminato quando il pod viene eliminato. Se l'amministratore del cluster ha creato diverse classi di archiviazione per il cluster, potrebbe essere possibile scegliere gli attributi dello storage da utilizzare, ad esempio se il volume viene eliminato o

recuperato dopo l'uso, se si espanderà se è necessario più spazio e anche se soddisfa determinati requisiti di prestazioni.

- Segreti: rendendo Secrets disponibili per i contenitori nelle specifiche del Pod, puoi fornire le autorizzazioni necessarie a tali contenitori per accedere a file system, database o altre risorse protette. Chiavi, password e token sono tra gli elementi che possono essere archiviati come segreti. L'uso dei segreti consente di non dover memorizzare queste informazioni nelle immagini dei container, ma è sufficiente rendere i segreti disponibili ai container in esecuzione. Simili a Secrets sono ConfigMaps. `ConfigMap` tende a contenere informazioni meno critiche, come coppie chiave-valore per la configurazione di un servizio.
- Risorse del container: gli oggetti per un'ulteriore configurazione dei container possono assumere la forma di configurazione delle risorse. Per ogni container, puoi richiedere la quantità di memoria e CPU che può utilizzare, nonché porre limiti alla quantità totale di tali risorse che il container può utilizzare. Per degli esempi, consultare Resource Management for Pods and Containers.
- Interruzioni: i pod possono essere interrotti involontariamente (un nodo non funziona) o volontariamente (è necessario un aggiornamento). Configurando un budget per le interruzioni dei pod, puoi esercitare un certo controllo sulla disponibilità dell'applicazione in caso di interruzioni. Per degli esempi, consultare Specifica di un budget di interruzioni per la tua applicazione.
- Namespace: Kubernetes offre diversi modi per isolare i componenti e i carichi di lavoro di Kubernetes gli uni dagli altri. L'esecuzione di tutti i pod per una particolare applicazione nello stesso namespace è un modo comune per proteggere e gestire insieme tali pod. Puoi creare i tuoi namespace da utilizzare o scegliere di non indicare un namespace (il che fa sì che Kubernetes utilizzi il namespace `default`). I componenti del piano di controllo di Kubernetes vengono in genere eseguiti nel namespace kube-system.

La configurazione appena descritta viene in genere raccolta in un file YAML da applicare al cluster Kubernetes. Per i cluster Kubernetes personali, è possibile archiviare semplicemente questi file YAML sul proprio sistema locale. Tuttavia, con cluster e carichi di lavoro più critici, GitOps è un modo popolare per automatizzare lo storage e gli aggiornamenti dei carichi di lavoro e delle risorse dell'infrastruttura Kubernetes.

Gli oggetti utilizzati per raccogliere e implementare le informazioni sui pod sono definiti da uno dei seguenti metodi di implementazione.

Implementazione di pod

Il metodo da scegliere per implementare i pod dipende dal tipo di applicazione che intendi eseguire con gli stessi. Ecco alcune delle scelte:

- Applicazioni stateless: un'applicazione stateless non salva i dati della sessione di un client, quindi un'altra sessione non deve fare riferimento a ciò che è accaduto a una sessione precedente. In questo modo è più semplice sostituire i pod con altri nuovi se non funzionano bene o spostarli senza salvarne lo stato. Se stai eseguendo un'applicazione stateless (come un server web), puoi utilizzare una distribuzione per distribuire Pods e. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/ https://kubernetes.io/docs/concepts/workloads/pods/ ReplicaSets A ReplicaSet definisce quante istanze di un Pod si desidera eseguire contemporaneamente. Sebbene sia possibile eseguire ReplicaSet direttamente un Pod, è comune eseguire le repliche direttamente all'interno di una distribuzione, per definire quante repliche di un Pod devono essere eseguite contemporaneamente.
- Applicazioni stateful: un'applicazione stateful è un'applicazione in cui l'identità del pod e l'ordine in cui vengono avviati i pod sono importanti. Queste applicazioni necessitano di uno storage persistente che sia stabile e che debba essere distribuito e scalato in modo coerente. Per distribuire un'applicazione con stato in Kubernetes, puoi usare. StatefulSets Un esempio di applicazione che viene in genere eseguita come un è un database. StatefulSet All'interno di un StatefulSet, è possibile definire le repliche, il Pod e i relativi contenitori, i volumi di archiviazione da montare e le posizioni nel contenitore in cui vengono archiviati i dati. Vedi Run a Replicated Steful Application per un esempio di database distribuito come. StatefulSet
- Per-node applicazioni: ci sono momenti in cui desideri eseguire un'applicazione su ogni nodo del cluster Kubernetes. Ad esempio, il tuo data center potrebbe richiedere che ogni computer esegua un'applicazione di monitoraggio o un particolare servizio di accesso remoto. Per Kubernetes, puoi utilizzare DaemonSet a per assicurarti che l'applicazione selezionata venga eseguita su tutti i nodi del cluster.
- Applicazioni eseguite fino al completamento: ci sono alcune applicazioni che desideri eseguire per completare un'attività particolare. Ciò potrebbe includerne una che esegue report mensili sullo stato o elimina i vecchi dati. Un oggetto Job può essere utilizzato per configurare un'applicazione per l'avvio e l'esecuzione, quindi uscire al termine dell'attività. Un CronJob oggetto consente di configurare un'applicazione in modo che venga eseguita a un'ora, un minuto, un giorno del mese, un mese o un giorno della settimana specifici, utilizzando una struttura definita dal formato crontab di Linux.

Rendere le applicazioni accessibili dalla rete

Con le applicazioni spesso distribuite come un insieme di microservizi che si spostano in luoghi diversi, Kubernetes aveva bisogno di un modo per consentire a tali microservizi di potersi trovare l'un l'altro. Inoltre, per consentire ad altri di accedere a un'applicazione esterna al cluster Kubernetes,

quest'ultimo aveva bisogno di un modo per esporre tale applicazione su indirizzi e porte esterni. Queste funzionalità relative alla rete vengono eseguite rispettivamente con oggetti Servizi e In ingresso:

- Servizi: poiché un pod può spostarsi su nodi e indirizzi diversi, un altro pod che deve comunicare con il primo pod potrebbe avere difficoltà a localizzarlo. Per risolvere questo problema, Kubernetes ti consente di rappresentare un'applicazione come servizio. Con un servizio, puoi identificare un pod o un set di pod con un nome particolare, quindi indicare quale porta espone il servizio di quell'applicazione dal pod e quali porte potrebbe utilizzare un'altra applicazione per contattare quel servizio. Un altro pod all'interno di un cluster può semplicemente richiedere un servizio per nome e Kubernetes indirizzerà la richiesta alla porta appropriata per un'istanza del pod che esegue quel servizio.
- In ingresso: In ingresso è ciò che può rendere disponibili le applicazioni rappresentate dai servizi Kubernetes ai client esterni al cluster. Le funzionalità di base di In ingresso includono un sistema di bilanciatore del carico (gestito da In ingresso), il controller In ingresso e regole per il routing delle richieste dal controller al servizio. Con Kubernetes, puoi scegliere tra diversi controller in ingresso.

## Fasi successive

Comprendere i concetti di base di Kubernetes e il modo in cui si relazionano ad Amazon EKS ti aiuterà a navigare sia la Documentazione su Amazon Elastic Kubernetes Service che la Documentazione di Kubernetes per trovare le informazioni necessarie per gestire i cluster Amazon EKS e implementare i carichi di lavoro su tali cluster. Per iniziare a utilizzare Amazon EKS, scegli tra i seguenti:

- the section called "Crea cluster (eksctl)"
- the section called "Creazione di un cluster "
- the section called "Implementazione esemplificativa (Linux)"
- Gestione dei cluster

# Implementazione di cluster Amazon EKS in ambienti cloud e on-premises

## Comprensione delle opzioni di implementazione di Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) è un servizio Kubernetes completamente gestito che semplifica l'uso di Kubernetes nel cloud e negli ambienti on-premises.

Nel cloud, Amazon EKS automatizza la gestione dell'infrastruttura del cluster Kubernetes per il piano di controllo (control-plane) e i nodi di Kubernetes. Ciò è essenziale per la pianificazione dei container, la gestione della disponibilità delle applicazioni, il dimensionamento dinamica delle risorse, l'ottimizzazione dell'elaborazione, l'archiviazione dei dati del cluster e l'esecuzione di altre funzioni critiche. Con Amazon EKS, ottieni solide prestazioni, scalabilità, affidabilità e disponibilità dell' AWS infrastruttura, oltre a integrazioni native con servizi AWS di rete, sicurezza, archiviazione e osservabilità.

Per semplificare l'esecuzione di Kubernetes nei tuoi ambienti on-premises, puoi utilizzare gli stessi cluster, funzionalità e strumenti di Amazon EKS in the section called "Nodi" o Amazon EKS Hybrid Nodes sulla tua infrastruttura, oppure puoi usare Amazon EKS Anywhere per ambienti isolati autonomi.

## Amazon EKS nel cloud

Puoi usare Amazon EKS con elaborazione in AWS Regions, AWS Local Zones e AWS Wavelength Zones. Con Amazon EKS nel cloud, la sicurezza, la scalabilità e la disponibilità del piano di controllo Kubernetes sono completamente gestite da AWS in the Region. AWS Quando esegui applicazioni con elaborazione in AWS regioni, ottieni l'intera gamma di funzionalità di Amazon EKS, inclusa la modalità automatica di AWS Amazon EKS, che automatizza completamente la gestione dell'infrastruttura del cluster Kubernetes per elaborazione, storage e rete con un solo clic. AWS Quando esegui applicazioni con elaborazione in AWS Local Zones e AWS Wavelength Zones, puoi utilizzare i nodi autogestiti di Amazon EKS per connettere le istanze Amazon EC2 per il calcolo del cluster e utilizzare gli altri servizi disponibili in Local Zones e Wavelength Zones. AWS AWS AWS Per ulteriori informazioni, consulta AWS Local Zones features and AWS Wavelength Zones features.

| | Amazon EKS nelle AWS regioni | Amazon EKS in Local/Wavelength zone |
|---|---|---|
| Gestione del piano di controllo Kubernetes | Gestito da AWS | Gestito da AWS |
| Posizione del piano di controllo Kubernetes | AWS Regioni | AWS Regioni |
| Piano dati di Kubernetes | • Modalità automatica di Amazon EKS<br>• Gruppi di nodi gestiti Amazon EKS<br>• Nodi di Amazon EC2 autogestiti<br>• AWS Fargate | • Gruppi di nodi gestiti di Amazon EKS (solo Local Zones)<br>• Nodi di Amazon EC2 autogestiti |
| Posizione del piano dati Kubernetes | AWS Regioni | AWS Zone locali o Wavelength |

## Amazon EKS nei tuoi data center o ambienti edge

Se devi eseguire applicazioni nei tuoi data center o ambienti edge, puoi utilizzare Amazon EKS su AWS Outposts o Amazon EKS Hybrid Nodes. Puoi utilizzare nodi autogestiti con istanze Amazon EC2 AWS su Outposts per il calcolo del cluster oppure puoi utilizzare Amazon EKS Hybrid Nodes con la tua infrastruttura locale o perimetrale per l'elaborazione del cluster. AWS Outposts è un'infrastruttura AWS gestita che gestisci nei tuoi data center o nelle strutture di co-location, mentre Amazon EKS Hybrid Nodes funziona su macchine fisiche o virtuali gestite in ambienti locali o periferici. Amazon EKS on AWS Outposts e Amazon EKS Hybrid Nodes richiedono una connessione affidabile dagli ambienti locali a una AWS regione e puoi utilizzare gli stessi cluster, funzionalità e strumenti Amazon EKS che usi per eseguire applicazioni nel cloud. Quando esegui su AWS Outposts, puoi in alternativa distribuire l'intero cluster Kubernetes su Outposts con i cluster locali Amazon EKS su AWS Outposts. AWS

| | Amazon EKS Hybrid Nodes | Amazon EKS su AWS Outposts |
|---|---|---|
| Gestione del piano di controllo Kubernetes | Gestito da AWS | Gestito da AWS |
| Posizione del piano di controllo Kubernetes | AWS Regioni | AWS Regioni o AWS Outposts |
| Piano dati di Kubernetes | Macchine fisiche o virtuali gestite dal cliente | Nodi di Amazon EC2 autogesti ti |
| Posizione del piano dati Kubernetes | Data center o ambiente edge del cliente | Data center o ambiente edge del cliente |

## Amazon EKS Anywhere per ambienti isolati

Amazon EKS Anywhere semplifica la gestione dei cluster Kubernetes attraverso l'automazione di operazioni indifferenziate come la configurazione dell'infrastruttura e le operazioni del ciclo di vita del cluster Kubernetes in ambienti on-premises ed edge. A differenza di Amazon EKS, Amazon EKS Anywhere è un prodotto gestito dai clienti e questi ultimi sono responsabili delle operazioni del ciclo di vita dei cluster e della manutenzione dei cluster Amazon EKS Anywhere. Amazon EKS Anywhere è basato sul sottoprogetto Kubernetes Cluster API (CAPI) e supporta una vasta gamma di infrastrutture tra cui VMware vSphere, bare metal, Nutanix, Apache e Snow. CloudStack AWS Amazon EKS Anywhere può essere eseguito in ambienti airgapped e offre integrazioni opzionali con AWS servizi regionali per l'osservabilità e la gestione delle identità. Per ricevere supporto per Amazon EKS Anywhere e accedere ai componenti aggiuntivi Kubernetes AWS forniti, puoi acquistare gli abbonamenti Amazon EKS Anywhere Enterprise.

| | Amazon EKS Anywhere |
|---|---|
| Gestione del piano di controllo Kubernetes | Gestito dal cliente |
| Posizione del piano di controllo Kubernetes | Data center o ambiente edge del cliente |
| Piano dati di Kubernetes | Macchine fisiche o virtuali gestite dal cliente |

| | Amazon EKS Anywhere |
|---|---|
| Posizione del piano dati Kubernetes | Data center o ambiente edge del cliente |

## Strumenti di Amazon EKS

Puoi utilizzare Amazon EKS Connector per registrare e connettere qualsiasi cluster Kubernetes conforme AWS e visualizzarlo nella console Amazon EKS. Una volta connesso, è possibile visualizzare lo stato, la configurazione e i carichi di lavoro del cluster nella console Amazon EKS. Puoi utilizzare questa funzionalità per visualizzare i cluster connessi nella console Amazon EKS, ma Amazon EKS Connector non semplifica la gestione o le operazioni di modifica dei cluster connessi tramite tale console.

Amazon EKS Distro è la AWS distribuzione dei componenti Kubernetes sottostanti che alimentano tutte le offerte Amazon EKS. Include i componenti principali necessari per un cluster Kubernetes funzionante, come i componenti del piano di controllo Kubernetes (etcd, kube-apiserver, kube-scheduler) e i componenti di rete (CoredNS, kube-proxy, plugin CNI). kube-controller-manager Amazon EKS Distro può essere utilizzato per gestire autonomamente i cluster Kubernetes con strumenti a tua scelta. Le distribuzioni di Amazon EKS Distro non sono coperte dai piani di supporto AWS .

# Configurazione per l'utilizzo di Amazon EKS

Per prepararti per la gestione a riga di comando dei cluster Amazon EKS, devi installare molti strumenti. Utilizza quanto segue per configurare le credenziali, creare e modificare i cluster e lavorare con i cluster una volta che sono in esecuzione:

- Configura AWS CLI: ottieni la AWS CLI per configurare e gestire i servizi necessari per lavorare con i cluster Amazon EKS. In particolare, è necessaria la AWS CLI per configurare le credenziali, ma è necessaria anche con altri servizi. AWS
- Configura kubectl ed eksctl: la `eksctl` CLI interagisce per creare, modificare ed eliminare i cluster Amazon AWS EKS. Una volta che un cluster è attivo, utilizza il comando `kubectl` open-source per gestire gli oggetti Kubernetes all'interno dei cluster Amazon EKS.
- Configura un ambiente di sviluppo (opzionale): valuta la possibilità di aggiungere i seguenti strumenti:
  - Strumento di implementazione locale: se non hai esperienza con Kubernetes, valuta la possibilità di installare uno strumento di implementazione locale come minikube o kind. Questi strumenti ti consentono di gestire un cluster Amazon EKS sulla macchina locale per testare le applicazioni.
  - Package manager — Helm è un popolare gestore di pacchetti per Kubernetes che semplifica l'installazione e la gestione di pacchetti complessi. Con Helm, è più facile installare e gestire pacchetti come Load AWS Balancer Controller sul tuo cluster Amazon EKS.

## Fasi successive

- Configura la CLI AWS
- Configura kubectl ed eksctl
- Avvio rapido: implementa un'app Web e archivia i dati

## Configurazione AWS CLI

La AWS CLI è uno strumento a riga di comando per lavorare con i AWS servizi, incluso Amazon EKS. Viene anche utilizzato per autenticare utenti o ruoli IAM per l'accesso al cluster Amazon EKS e ad altre AWS risorse dal computer locale. Per effettuare il provisioning delle risorse AWS dalla riga di comando, è necessario ottenere un ID della chiave di AWS accesso e una chiave segreta da utilizzare nella riga di comando. Quindi è necessario configurare queste credenziali nella AWS CLI.

Se non hai già installato la AWS CLI, vedi Installare o aggiornare la versione più recente della AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

## Per creare una chiave di accesso

1. Accedi alla Console di gestione AWS.
2. Per account con utente singolo o multiplo:
   - Single-user account —:: In alto a destra, scegli il tuo nome AWS utente per aprire il menu di navigazione. Ad esempio, scegli **`webadmin`**.
   - Multiple-user account —: Scegli IAM dall'elenco dei servizi. Dalla dashboard IAM, seleziona Utenti e scegli il nome dell'utente.
3. Scegli Security Credentials (Credenziali di sicurezza).
4. In Chiavi di accesso, seleziona Crea chiave di accesso.
5. Scegli Interfaccia a riga di comando (CLI), quindi scegli Avanti.
6. Selezionare Create access key (Crea chiave di accesso).
7. Scegli Scarica file .csv.

## Per configurare AWS CLI

Dopo aver installato la AWS CLI, effettuate le seguenti operazioni per configurarla. Per ulteriori informazioni, consulta Configurare la AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

1. In una finestra del terminale, digita il comando riportato qui sotto:

   ```
   aws configure
   ```

   Facoltativamente, puoi configurare un profilo con un nome, ad esempio `--profile cluster-admin`. Se configuri un profilo denominato nella AWS CLI, devi sempre passare questo flag nei comandi successivi.
2. Inserisci le tue AWS credenziali. Ad esempio:

   ```
   Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [None]: region-code
   ```

```
Default output format [None]: json
```

## Per ottenere un token di sicurezza

Se necessario, esegui il comando seguente per ottenere un nuovo token di sicurezza per la AWS CLI. Per ulteriori informazioni, vedete get-session-token nella CLI Command Reference. AWS

Per impostazione predefinita, il token è valido per 15 minuti. Per modificare il timeout di sessione predefinito, passa il flag `--duration-seconds`. Ad esempio:

```
aws sts get-session-token --duration-seconds 3600
```

Questo comando restituisce le credenziali di sicurezza temporanee per una sessione CLI. AWS Dovresti vedere il seguente messaggio di risposta:

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

## Per verificare l'identità dell'utente

Se necessario, esegui il comando seguente per verificare le AWS credenziali della tua identità utente IAM (ad esempio*ClusterAdmin*) per la sessione del terminale.

```
aws sts get-caller-identity
```

Questo comando restituisce l'Amazon Resource Name (ARN) dell'entità IAM configurata per la CLI AWS . Dovresti vedere il seguente esempio di risposta:

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws: iam::01234567890:user/ClusterAdmin"
}
```

## Fasi successive

- Configura kubectl ed eksctl
- Avvio rapido: implementa un'app Web e archivia i dati

# Configura  kubectl ed eksctl

> **Tip**
>
> Registrati  per i prossimi workshop Amazon EKS.

Una volta installata la AWS CLI, ci sono altri due strumenti da installare per creare e gestire i cluster Kubernetes:

- `kubectl`: lo strumento a riga di comando `kubectl` è lo strumento principale per gestire le risorse all'interno del tuo cluster Kubernetes. Questa pagina descrive come scaricare e configurare il file binario `kubectl` che corrisponde alla versione del cluster Kubernetes. Consultare Install or update kubectl.
- `eksctl`: lo strumento a riga di `eksctl` comando è progettato per creare cluster EKS nel AWS cloud o in locale (con EKS Anywhere), nonché per modificare ed eliminare tali cluster. Consultare Installare eksctl.

## `Installa o aggiorna kubectl`

In questa sezione viene descritto come scaricare e installare o aggiornare il file binario `kubectl` sul tuo dispositivo. Il file binario è identico alle versioni della community upstream. Il file binario non è esclusivo di Amazon EKS o. AWS Segui i passaggi seguenti per ottenere la versione specifica di `kubectl` cui hai bisogno, anche se molti builder si limitano a eseguire `brew install kubectl` per installarla.

> **Note**
>
> Utilizza la versione secondaria `kubectl` immediatamente precedente a quella del piano di controllo del cluster Amazon EKS. Ad esempio, un client `1.35` `kubectl` deve funzionare con cluster Kubernetes `1.34`, `1.35` e `1.36`.

> **Tip**
>
> Puoi anche `kubectl` utilizzarlo direttamente dalla console EKS tramite AWS CloudShell. Scegli  Connetti  su qualsiasi pagina dei dettagli del cluster per ottenere una sessione preconfigurata. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".

## Passaggio 1: controlla se  kubectl è installato

Determina se `kubectl` è già installato sul tuo dispositivo.

```
kubectl version --client
```

Se `kubectl` è già installato nel percorso del tuo dispositivo, l'output di esempio include informazioni simili alle seguenti. Se desideri aggiornare la versione correntemente installata con una versione più recente, completa la fase successiva assicurandoti di installare la nuova versione nella stessa posizione in cui si trova la versione corrente.

```
Client Version: v1.31.X-eks-1234567
```

Se non ricevi alcun output, allora `kubectl` non è installato o è installato in una posizione che non si trova nel percorso del dispositivo.

## Passaggio 2: installa o aggiorna kubectl

Installare o aggiornare `kubectl` su uno dei seguenti sistemi operativi:

- the section called "macOS"
- the section called "Linux (amd64)"
- the section called "Linux (arm64)"
- the section called "Windows"

> **Note**
>
> Se i download AWS nella tua regione sono lenti dalle AWS regioni utilizzate in questa sezione, valuta la possibilità di impostare la visualizzazione in primo piano dei CloudFront

contenuti. Per ulteriori informazioni, consulta Guida introduttiva a una CloudFront distribuzione di base.

## macOS

Segui la seguente procedura per installare `kubectl` su macOS. Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Aggiungi l'esecuzione alle autorizzazioni del file binario.
- Copia il file binario in una cartella nel tuo PATH.
- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Scarica il file binario per la versione del cluster di Kubernetes da Amazon S3.
   - Kubernetes `1.36`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.35`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/
darwin/amd64/kubectl
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/
darwin/amd64/kubectl
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/
darwin/amd64/kubectl
```

2. (Facoltativo) Verifica il file binario scaricato con il relativo checksum `SHA-256`.
   a. Scarica il checksum di `SHA-256` per la versione del tuo cluster di Kubernetes.
      - Kubernetes `1.36`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.35`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.34`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/darwin/amd64/kubectl.sha256
```

b. Controlla il checksum `SHA-256` del file binario scaricato.

```
openssl sha1 -sha256 kubectl
```

c. Assicurati che il checksum generato nell'output corrisponda al checksum nel file scaricato. `kubectl.sha256`

3. Applica le autorizzazioni di esecuzione al file binario.

```
chmod +x ./kubectl
```

4. Copia il file binario in una cartella nel tuo `PATH`. Se disponi già di una versione installata di `kubectl`, consigliamo di creare un `$HOME/bin/kubectl` e verificare che `$HOME/bin` venga per primo in `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facoltativo) Aggiungi il percorso `$HOME/bin` al file di inizializzazione dello shell (interprete di comandi), in modo che sia configurato all'apertura di quest'ultimo.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

## Linux (amd64)

Segui la seguente procedura per installare `kubectl` su Linux (amd64). Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Aggiungi l'esecuzione alle autorizzazioni del file binario.

- Copia il file binario in una cartella nel tuo PATH.
- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Scarica il file binario `kubectl` per la versione del cluster di Kubernetes da Amazon S3.
   - Kubernetes `1.36`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.35`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/linux/amd64/kubectl
     ```

2. (Facoltativo) Verifica il file binario scaricato con il relativo checksum SHA-256.

   a. Scarica il SHA-256 checksum per la versione Kubernetes del tuo cluster da Amazon S3 utilizzando il comando per la piattaforma hardware del tuo dispositivo.

      - Kubernetes 1.36

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/linux/amd64/kubectl.sha256
      ```

      - Kubernetes 1.35

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/linux/amd64/kubectl.sha256
      ```

      - Kubernetes 1.34

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/linux/amd64/kubectl.sha256
      ```

      - Kubernetes 1.33

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/linux/amd64/kubectl.sha256
      ```

      - Kubernetes 1.32

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/linux/amd64/kubectl.sha256
      ```

      - Kubernetes 1.31

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/linux/amd64/kubectl.sha256
      ```

      - Kubernetes 1.30

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/linux/amd64/kubectl.sha256
      ```

   b. Controlla la somma di controllo SHA-256 del file binario scaricato con uno dei seguenti comandi.

```
sha256sum -c kubectl.sha256
```

or

```
openssl sha1 -sha256 kubectl
```

c. Per il primo, dovresti vederekubectl: OK, per il secondo, puoi verificare che il checksum generato nell'output corrisponda al checksum nel file scaricato. kubectl.sha256

3. Applica le autorizzazioni di esecuzione al file binario.

```
chmod +x ./kubectl
```

4. Copia il file binario in una cartella nel tuo PATH. Se disponi già di una versione installata di kubectl, consigliamo di creare un $HOME/bin/kubectl e verificare che $HOME/bin venga per primo in $PATH.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facoltativo) Aggiungi il percorso $HOME/bin al file di inizializzazione dello shell (interprete di comandi), in modo che sia configurato all'apertura di quest'ultimo.

> **Note**
>
> Questa fase prevede l'utilizzo dello shell (interprete di comandi) Bash; se stai utilizzando un altro shell, modifica il comando per utilizzare il file di inizializzazione dello shell in uso.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

## Linux (arm64)

Segui la seguente procedura per installare kubectl su Linux (arm64). Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Aggiungi l'esecuzione alle autorizzazioni del file binario.

- Copia il file binario in una cartella nel tuo PATH.
- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Scarica il file binario `kubectl` per la versione del cluster di Kubernetes da Amazon S3.
   - Kubernetes `1.36`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.35`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/linux/arm64/kubectl
     ```

2. (Facoltativo) Verifica il file binario scaricato con il relativo checksum SHA-256.

   a. Scarica il SHA-256 checksum per la versione Kubernetes del tuo cluster da Amazon S3 utilizzando il comando per la piattaforma hardware del tuo dispositivo.

      - Kubernetes 1.36

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/linux/arm64/kubectl.sha256
      ```

      - Kubernetes 1.35

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/linux/arm64/kubectl.sha256
      ```

      - Kubernetes 1.34

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/linux/arm64/kubectl.sha256
      ```

      - Kubernetes 1.33

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/linux/arm64/kubectl.sha256
      ```

      - Kubernetes 1.32

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/linux/arm64/kubectl.sha256
      ```

      - Kubernetes 1.31

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/linux/arm64/kubectl.sha256
      ```

      - Kubernetes 1.30

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/linux/arm64/kubectl.sha256
      ```

   b. Controlla la somma di controllo SHA-256 del file binario scaricato con uno dei seguenti comandi.

```
sha256sum -c kubectl.sha256
```

or

```
openssl sha1 -sha256 kubectl
```

c. Per il primo, dovresti vederekubectl: OK, per il secondo, puoi verificare che il checksum generato nell'output corrisponda al checksum nel file scaricato. kubectl.sha256

3. Applica le autorizzazioni di esecuzione al file binario.

```
chmod +x ./kubectl
```

4. Copia il file binario in una cartella nel tuo PATH. Se disponi già di una versione installata di kubectl, consigliamo di creare un $HOME/bin/kubectl e verificare che $HOME/bin venga per primo in $PATH.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facoltativo) Aggiungi il percorso $HOME/bin al file di inizializzazione dello shell (interprete di comandi), in modo che sia configurato all'apertura di quest'ultimo.

> **Note**
>
> Questa fase prevede l'utilizzo dello shell (interprete di comandi) Bash; se stai utilizzando un altro shell, modifica il comando per utilizzare il file di inizializzazione dello shell in uso.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

## Windows

Segui la seguente procedura per installare kubectl su Windows. Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Copia il file binario in una cartella nel tuo PATH.

- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Aprire un terminale PowerShell .
2. Scarica il file binario `kubectl` per la versione del cluster di Kubernetes da Amazon S3.
   - Kubernetes `1.36`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.35`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.34`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.33`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.32`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.31`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.30`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/windows/amd64/kubectl.exe
     ```

3. (Facoltativo) Verifica il file binario scaricato con il relativo checksum SHA-256.
   a. Scarica il checksum di SHA-256 per la versione per Windows del tuo cluster di Kubernetes.
      - Kubernetes 1.36

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.36.2/2026-07-05/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes 1.35

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.35.6/2026-07-05/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes 1.34

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.9/2026-07-05/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes 1.33

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.13/2026-07-05/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes 1.32

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.13/2026-07-05/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes 1.31

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.14/2026-07-05/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes 1.30

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2026-07-05/bin/windows/amd64/kubectl.exe.sha256
```

   b. Controlla il checksum SHA-256 del file binario scaricato.

```
Get-FileHash kubectl.exe
```

c. Assicurati che il checksum generato nell'output corrisponda al checksum nel file scaricato. `kubectl.exe.sha256` L' PowerShell output deve essere una stringa di caratteri equivalente in lettere maiuscole.

4. Copia il file binario in una cartella nel tuo `PATH`. Se disponi di una directory esistente nel `PATH`dedicata alle utility a riga di comando, copia il file binario in questa directory. In alternativa, completa la procedura seguente.
   a. Crea una nuova directory per i file binari della riga di comando, ad esempio `C:\bin`.
   b. Copia il file binario `kubectl.exe` nella nuova directory.
   c. Modifica la variabile di ambiente `PATH` dell'utente o del sistema per aggiungere la nuova directory a `PATH`.
   d. Chiudi il PowerShell terminale e apri uno nuovo per selezionare la nuova `PATH` variabile.
5. Dopo l'installazione di `kubectl`, puoi verificarne la versione.

```
kubectl version --client
```

6. Alla prima installazione, `kubectl` non è ancora configurato per comunicare con alcun server. Tratteremo questa configurazione, se necessario, in altre procedure. Per aggiornare la configurazione per comunicare con un particolare cluster, è possibile eseguire il comando seguente. Sostituiscila *`region-code`* con la AWS regione in cui si trova il cluster. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

7. Prendi in considerazione la configurazione del completamento automatico, che consente di utilizzare il tasto TAB per completare i sottocomandi `kubectl` dopo aver digitato le prime lettere. Per i dettagli, consultare Kubectl autocomplete nella documentazione di Kubernetes.

## Installa **`eksctl`**

`eksctl` CLI viene utilizzata per lavorare con i cluster EKS. Automatizza molte attività individuali. Per istruzioni sull'installazione di `eksctl`, consultare Installation nella documentazione di `eksctl`. Per Linux, usa le istruzioni UNIX.

Durante l'utilizzo`eksctl`, il responsabile di sicurezza IAM che stai utilizzando deve disporre delle autorizzazioni per lavorare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi AWS CloudFormation, un VPC e le risorse correlate. Per ulteriori informazioni, consulta le pagine Actions

e Using service-linked roles nella Guida per l'utente IAM. È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:

```
aws sts get-caller-identity
```

## Fasi successive

- Avvio rapido: implementa un'app Web e archivia i dati

# Avvio rapido: implementazione di un'app Web e archiviazione dei dati

Questo tutorial di avvio rapido ti guida attraverso i passaggi per distribuire l'applicazione di esempio di gioco 2048 e mantenere i suoi dati su un cluster Amazon EKS Auto Mode utilizzando eksctl.

La modalità automatica di Amazon EKS semplifica la gestione dei cluster automatizzando attività di routine come lo storage a blocchi, il networking, il bilanciamento del carico e la scalabilità automatica del calcolo. Durante la configurazione, gestisce la creazione di nodi con istanze gestite EC2, sistemi di bilanciamento del carico delle applicazioni e volumi EBS.

In sintesi, distribuirai un carico di lavoro di esempio con le annotazioni personalizzate necessarie per una perfetta integrazione con i servizi. AWS

## In questo tutorial

Utilizzando il modello di cluster `eksctl` che segue, creerai un cluster con modalità automatica EKS per il provisioning automatico dei nodi.

- Configurazione VPC: quando si utilizza il modello di cluster eksctl che segue, eksctl crea automaticamente un Virtual Private Cloud (VPC) IPv4 per il cluster. Per impostazione predefinita, eksctl configura un VPC che soddisfa tutti i requisiti di rete, oltre a creare endpoint pubblici e privati.
- Gestione delle istanze: EKS Auto Mode aggiunge o rimuove dinamicamente i nodi nel cluster EKS in base alle esigenze delle applicazioni Kubernetes.
- Persistenza dei dati: utilizzate la funzionalità di archiviazione a blocchi di EKS Auto Mode per garantire la persistenza dei dati delle applicazioni, anche in scenari che prevedono riavvii o guasti dei pod.
- Accesso esterno alle app: utilizza la funzionalità di bilanciamento del carico di EKS Auto Mode per effettuare il provisioning dinamico di un Application Load Balancer (ALB).

## Prerequisiti

Prima di iniziare, esegui le seguenti attività:

- Configura il tuo ambiente per Amazon EKS

- Installa la versione più recente di eksctl

## Configurazione del cluster

In questa sezione creerai un cluster utilizzando la modalità automatica EKS per il provisioning dinamico dei nodi.

Crea un file `cluster-config.yaml` e incollaci il seguente contenuto. Sostituisci `region-code` con una regione valida (ad es.). `us-east-1`

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: web-quickstart
  region: <region-code>

autoModeConfig:
  enabled: true
```

Ora siamo pronti per creare il cluster.

Crea il cluster EKS utilizzando`cluster-config.yaml`:

```
eksctl create cluster -f cluster-config.yaml
```

> **Important**
>
> Se non usi eksctl per creare il cluster, devi etichettare manualmente le sottoreti VPC.

## Crea IngressClass

Crea un Kubernetes `IngressClass` per la modalità automatica EKS. IngressClass Definisce come EKS Auto Mode gestisce le risorse di Ingress. Questo passaggio configura la capacità di bilanciamento del carico della modalità automatica EKS. Quando create risorse Ingress per le vostre applicazioni, EKS Auto Mode le utilizza per fornire e IngressClass gestire automaticamente i sistemi di bilanciamento del carico, integrando le applicazioni Kubernetes con i servizi di bilanciamento del carico. AWS

Salva il seguente file yaml come `ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: eks.amazonaws.com/alb
```

Applica al tuo cluster: IngressClass

```
kubectl apply -f ingressclass.yaml
```

## Distribuzione dell'applicazione di gioco 2048 esemplificativa

In questa sezione, ti illustreremo i passaggi per implementare il popolare "gioco 2048" come applicazione esemplificativa all'interno del cluster. Il manifesto fornito include annotazioni personalizzate per l'Application Load Balancer (ALB). Queste annotazioni si integrano e indicano a EKS di gestire il traffico HTTP in entrata come «rivolto a Internet» e di indirizzarlo al servizio appropriato nel `game-2048` namespace utilizzando il tipo di destinazione «ip».

> **Note**
>
> L'immagine `docker-2048` nell'esempio è un'immagine di container `x86_64` e non sarà eseguita su altre architetture.

1. Crea un namespace Kubernetes chiamato `game-2048` con il flag `--save-config`.

   ```
   kubectl create namespace game-2048 --save-config
   ```

   Dovresti vedere il seguente messaggio di risposta:

   ```
   namespace/game-2048 created
   ```

2. Implementare l'applicazione di gioco 2048 esemplificativa.

```
kubectl apply -n game-2048 -f https://raw.githubusercontent.com/kubernetes-sigs/aws-
load-balancer-controller/v2.8.0/docs/examples/2048/2048_full.yaml
```

Questo manifesto configura l'implementazione, il servizio e l'entrata Kubernetes per il namespace `game-2048`, creando le risorse necessarie per implementare ed esporre l'applicazione `game-2048` all'interno del cluster. Include la creazione di un servizio denominato `service-2048` che espone l'implementazione sulla porta `80` e una risorsa di entrata denominata `ingress-2048` che definisce le regole di routing per il traffico HTTP in entrata e le annotazioni per un Application Load Balancer (ALB) con accesso a Internet. Dovresti vedere il seguente messaggio di risposta:

```
namespace/game-2048 configured
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
```

3. Esegui il seguente comando per ottenere la risorsa di ingresso per il namespace `game-2048`.

```
kubectl get ingress -n game-2048
```

Dovresti vedere il seguente messaggio di risposta:

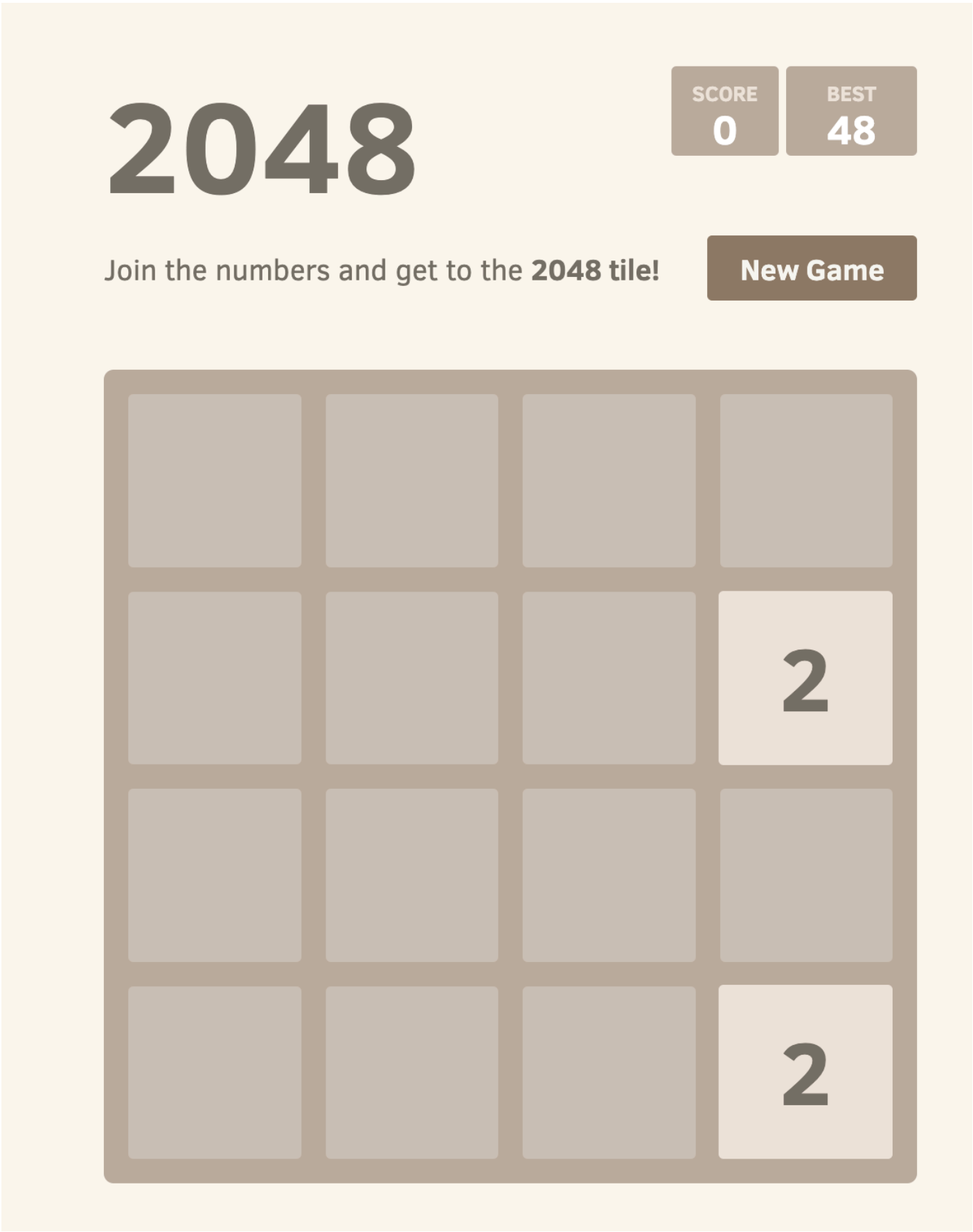
```
NAME           CLASS   HOSTS   ADDRESS
                      PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-eb379a0f83-378466616.region-
code.elb.amazonaws.com   80      31s
```

Dovrai attendere alcuni minuti per effettuare il provisioning dell'Application Load Balancer (ALB) prima di iniziare con i passaggi seguenti.

4. Apri un browser Web e inserisci `ADDRESS` dal passaggio precedente per accedere all'applicazione Web. Esempio:

```
k8s-game2048-ingress2-eb379a0f83-378466616.region-code.elb.amazonaws.com
```

Il gioco 2048 dovrebbe essere visualizzato nel browser in uso. Gioca!




Distribuzione dell'applicazione di gioco 2048 esemplificativa

# Dati persistenti utilizzando la modalità automatica Amazon EKS

Ora che il gioco 2048 è attivo e funzionante sul tuo cluster Amazon EKS, è il momento di garantire che i tuoi dati di gioco siano conservati in modo sicuro, utilizzando la funzionalità di archiviazione a blocchi della modalità automatica Amazon EKS.

1. Creare un file denominato `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Applica `StorageClass`:

```
kubectl apply -f storage-class.yaml
```

3. Crea una richiesta di volumi persistenti (PVC) per richiedere lo spazio di archiviazione per i tuoi dati di gioco. Crea un file denominato `ebs-pvc.yaml` che contenga quando segue e aggiungilo a .

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: game-data-pvc
  namespace: game-2048
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: auto-ebs-sc
```

4. Applica PVC al cluster:

```
kubectl apply -f ebs-pvc.yaml
```

Dovresti vedere il seguente messaggio di risposta:

```
persistentvolumeclaim/game-data-pvc created
```

5. Ora devi aggiornare l'implementazione del gioco 2048 per utilizzare questo PVC per archiviare i dati. La seguente implementazione è configurata per utilizzare PVC per l'archiviazione dei dati di gioco. Crea un file denominato `ebs-deployment.yaml` e aggiungilo a esso:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3  # Adjust the number of replicas as needed
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          volumeMounts:
            - name: game-data
              mountPath: /var/lib/2048
      volumes:
        - name: game-data
          persistentVolumeClaim:
            claimName: game-data-pvc
```

6. Applica l'implementazione aggiornata:

```
kubectl apply -f ebs-deployment.yaml
```

Dovresti vedere il seguente messaggio di risposta:

```
deployment.apps/deployment-2048 configured
```

Con questi passaggi, il gioco 2048 sul cluster è ora configurato per rendere persistenti i dati, utilizzando la funzionalità di archiviazione a blocchi della modalità automatica Amazon EKS. Ciò garantisce che i progressi e i dati di gioco siano al sicuro anche in caso di guasti ai pod o ai nodi.

Se ti è piaciuto questo tutorial, faccelo sapere fornendo un feedback, così saremo in grado di fornirti altri tutorial rapidi di utilizzo specifici per ogni caso, come questo.

## Eliminazione

Per evitare di incorrere in addebiti futuri, è necessario eliminare manualmente lo CloudFormation stack associato per eliminare tutte le risorse create durante questa guida, inclusa la rete VPC.

Eliminare lo stack: CloudFormation

```
eksctl delete cluster -f ./cluster-config.yaml
```

# Scopri Amazon EKS con l'esempio

## Panoramica di

Questa guida per l'utente di Amazon EKS contiene procedure generiche per creare il tuo primo cluster EKS dalla riga di comando o Console di gestione AWS e un solido riferimento per tutti i principali componenti di Amazon EKS. Tuttavia, in qualità di amministratore o sviluppatore di cluster Amazon EKS, puoi acquisire una comprensione più approfondita di Amazon EKS seguendo i percorsi di apprendimento presenti in siti esterni a questa guida. Questi siti possono aiutarti a:

- Configurare tipi di cluster specifici. I tipi di cluster specifici possono essere basati sui tipi di carico di lavoro o sui requisiti di sicurezza. Ad esempio, potresti voler ottimizzare un cluster per eseguire carichi di lavoro in batch, di machine learning o con elaborazione intensiva.
- Migliorare i tuoi cluster. Puoi aggiungere funzionalità avanzate al tuo cluster per fornire elementi come osservabilità, archiviazione flessibile, scalabilità automatica o reti di cluster specializzate.
- Automatizzare gli aggiornamenti. Utilizzando funzionalità come GitOps, puoi configurare il provisioning automatico dell'infrastruttura del cluster e dei carichi di lavoro, in base alle modifiche che si verificano a tali componenti nei tuoi repository Git.
- Utilizzare strumenti avanzati di configurazione del cluster. Anche se `eksctl` rappresenta un modo rapido per creare un cluster, esistono altri strumenti che possono semplificare la configurazione e l'aggiornamento di cluster più complessi. Questi includono strumenti come Terraform e. CloudFormation

Per iniziare il tuo percorso di apprendimento su Amazon EKS, ti consigliamo di visitare alcuni dei siti descritti in questa pagina. Se riscontri problemi lungo il percorso, sono disponibili anche delle risorse per aiutarti a risolverli. Ad esempio, il Re:post Knowledge Center ti consente di cercare nel database di supporto i problemi relativi al EKS-related supporto di Amazon. Inoltre, la Amazon EKS Best Practices Guide offre suggerimenti sui modi migliori per configurare cluster di livello di produzione.

## Workshop su Amazon EKS

Partendo da una conoscenza di base di Kubernetes e dei container, Amazon EKS workshop è una piattaforma di apprendimento per insegnare agli amministratori dei cluster le funzionalità importanti di Amazon EKS. Ecco alcuni modi per partecipare al workshop Amazon EKS:

- Nozioni di base su Amazon EKS: guarda il video nella pagina introduttiva per scoprire come Amazon EKS implementa le funzionalità di Kubernetes sul cloud. AWS Se hai bisogno di conoscere meglio le basi di Kubernetes, guarda il video What is Kubernetes.
- Configurazione di Amazon EKS: se disponi di un AWS account, la sezione Configurazione ti aiuta a configurare un CloudShell ambiente da utilizzare per creare un cluster. Infatti, offre una scelta tra eksctl (una semplice riga di comando per la creazione di cluster) e Terraform (un approccio più da infrastruttura come codice alla creazione di un cluster) per creare un cluster Amazon EKS.
- Amazon EKS Getting started: prova un semplice web store nella sezione Sample application. Puoi utilizzarlo durante gli altri esercizi. In questa sezione, puoi anche scoprire come impacchettare le immagini dei container e come i microservizi vengono gestiti utilizzando Kubernetes Pods, Deployments, Services e Namespaces. StatefulSets Quindi utilizza Kustomize per implementare le modifiche ai manifesti di Kubernetes.
- Nozioni fondamentali su Amazon EKS: utilizzando AWS funzionalità come il Load AWS Balancer Controller, il workshop mostra come esporre le applicazioni al mondo esterno. Per quanto riguarda lo storage, il workshop illustra come utilizzare Amazon EBS per lo storage a blocchi, Amazon EFS per lo storage di file system e Amazon FSx for ONTAP per gestire i file system ONTAP in. NetApp AWS Per la gestione dei nodi, il workshop ti aiuta a configurare i gruppi di nodi gestiti.
- Funzionalità avanzate di Amazon EKS: le funzionalità più avanzate offerte tramite il workshop Amazon EKS includono laboratori per la configurazione di:
  - Dimensionamento automatico: include il dimensionamento automatico dei nodi (con Cluster Autoscaler o Karpenter) e quello dei carichi di lavoro (con Horizontal Pod Autoscaler e Cluster Proportional Autoscaler).
  - Osservabilità: scopri di più sulla registrazione OpenSearch, sui container Insights su Amazon EKS e sulla visibilità dei costi con Kubecost in una serie di laboratori di osservabilità. https://www.eksworkshop.com/docs/observability/
  - Sicurezza: questo set di laboratori di sicurezza ti consente di esplorare Secrets Management, Amazon, Pod Security Standards e la gestione delle policy di GuardDuty Kyverno. https://www.eksworkshop.com/docs/security/kyverno/
  - Rete: scopri le funzionalità di rete per Amazon EKS dai Networking labs, tra cui Amazon VPC CNI (plugin di rete di supporto) e Amazon VPC Lattice (per configurare i cluster su VPC e account utente).
  - Automazione: Labs on Automation ti guida attraverso GitOps metodi di gestione di cluster e progetti come AWS Controllers for Kubernetes e Crossplane per la gestione dei piani di controllo di Amazon EKS. https://www.eksworkshop.com/docs/automation/controlplanes/crossplane/

# Tutorial pratici per la configurazione dei cluster su Amazon EKS

Una serie di tutorial sulla configurazione dei cluster Amazon EKS disponibili sul sito della AWS community può aiutarti a creare cluster Amazon EKS speciali e a migliorarli in vari modi. I tutorial sono suddivisi in tre tipi diversi:

Creazione di cluster

Questi tutorial aiutano a creare cluster che possono essere utilizzati per scopi speciali. Tra questi, c'è la possibilità di eseguire:

- Applicazioni scalabili a livello globale basate su IPv6
- Attività in batch asincrone
- Microservizi ad alto traffico
- Dimensionamento automatico con Karpenter su Fargate
- Carichi di lavoro finanziari
- Gruppi di nodi gestiti da Windows

Potenziamento dei cluster

Una volta che disponi di un cluster esistente, puoi estenderlo e migliorarlo in modo da consentirgli di eseguire carichi di lavoro specializzati e di potenziare in altro modo i cluster. Questi tutorial includono modi per:

- Fornire soluzioni di archiviazione con CSI EFS
- Fornire un'archiviazione dinamica del database con CSI EBS
- Esponi le applicazioni su cluster IPv4 utilizzando il Load Balancer Controller AWS
- Esporre le applicazioni su cluster IPv6 utilizzando il Load Balancer Controller AWS

AWS Ottimizzazione dei servizi

Utilizzando questi tutorial, puoi integrare meglio i tuoi cluster con i servizi. AWS Questi tutorial includono quelli che aiutano a:

- Gestire i record DNS per i microservizi con ExternalDNS
- Monitora le applicazioni con CloudWatch

- Gestire le attività asincrone con l'archiviazione SQS ed EFS
- Utilizza AWS Secrets Manager Secrets Secrets dai carichi di lavoro
- Configurare mTLs con Fargate, NGINX e ACM PCA

## Esempi di Amazon EKS

Il repository degli esempi di Amazon EKS memorizza i manifesti da utilizzare con Amazon EKS. Questi manifesti offrono l'opportunità di provare diversi tipi di applicazioni in Amazon EKS o creare tipi specifici di cluster Amazon EKS. Tra gli esempi troviamo manifesti per:

- Crea un cluster AWS Amazon EKS Fargate
- Creare un cluster con un ruolo IAM esistente
- Aggiungi un gruppo di nodi gestiti da Ubuntu a un cluster
- Eseguire il backup e ripristinare l'archiviazione dei pod con snapshot di volume
- Recuperare i volumi EBS montati come PVC con più account
- Abilitare il protocollo proxy per NGINX Ingress Controller con Classic Load Balancer
- Configura la registrazione su Fargate su AWS OpenSearch
- Eseguire SDK Python con un provider di identità federato web
- Implementare un'app di esempio su un controller CSI NFS
- Usa le istantanee del volume per StatefulSets
- Distribuire i pod tra nodi in zone di disponibilità diverse

Tieni presente che questi esempi sono solo a scopo di apprendimento e test e non sono destinati all'utilizzo in produzione.

## AWS Esercitazioni

Il sito AWS Tutorials pubblica alcuni tutorial di Amazon EKS, ma offre anche uno strumento di ricerca per trovare altri tutorial pubblicati su AWS siti (come il sito della community). AWS I tutorial di Amazon EKS pubblicati direttamente su questo sito includono:

- Distribuzione di un'app web per container su Amazon EKS
- Esecuzione di cluster Kubernetes a un prezzo inferiore (istanze Amazon EKS e Spot)
- Come ottimizzare i costi dei lavori Jenkins su Kubernetes

# Workshop per sviluppatori

Se sei uno sviluppatore di software e desideri creare o rifattorizzare applicazioni da eseguire su Amazon EKS, il workshop Amazon EKS Developers è un buon punto di partenza. Questo workshop non aiuta solo a creare applicazioni containerizzate, ma anche a distribuire tali container in un registro di container (ECR) e da lì in un cluster Amazon EKS.

Inizia con Amazon EKS Python Workshop per completare il processo di rifattorizzazione di un'applicazione Python, quindi configura il tuo ambiente di sviluppo per prepararti alla distribuzione dell'applicazione. Consulta le sezioni relative a container, Kubernetes e Amazon EKS per prepararti a eseguire le applicazioni containerizzate in quegli ambienti.

# Workshop Terraform

Anche se `eksctl` è uno strumento semplice per la creazione di un cluster, per tipi di implementazioni Amazon EKS di tipo infrastruttura come codice più complessi, Terraform è un popolare strumento di creazione e gestione di cluster Amazon EKS. Il workshop Terraform Amazon EKS insegna come utilizzare Terraform per creare un AWS VPC, creare cluster Amazon EKS e aggiungere miglioramenti opzionali al cluster. In particolare, esiste una sezione per la creazione di un cluster Amazon EKS privato

# AWS Formazione Amazon EKS

AWS offre una formazione formale per conoscere Amazon EKS. Un corso di formazione di tre giorni intitolato Running Containers on Amazon Elastic Kubernetes Service insegna:

- Nozioni di base su Kubernetes e Amazon EKS
- Come creare cluster Amazon EKS
- Protezione di Amazon EKS con l'autorizzazione AWS IAM e Kubernetes RBAC
- GitOps strumenti di automazione
- strumenti di monitoraggio
- Tecniche per migliorare costi, efficienza e resilienza

# Nozioni di base su Amazon EKS

Prima di consultare le guide introduttive, verifica di aver completato la configurazione necessaria per utilizzare Amazon EKS. Per ulteriori informazioni, consulta Configurazione.

Sono disponibili due guide di istruzioni rapide per la creazione di un nuovo cluster Kubernetes con nodi in Amazon EKS:

- Nozioni di base su Amazon EKS – eksctl: questa guida alle operazioni di base consente di installare tutte le risorse necessarie per iniziare a usare Amazon EKS attraverso `eksctl`, una semplice utility a riga di comando per la creazione e la gestione di cluster di Kubernetes su Amazon EKS. Al termine del tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni. Questo è il modo più rapido e semplice per iniziare a utilizzare Amazon EKS.
- Nozioni di base su Amazon EKS – Console di gestione AWS e AWS CLI: questa guida alle operazioni di base ti consente di creare tutte le risorse necessarie per iniziare a utilizzare Amazon EKS attraverso Console di gestione AWS e AWS CLI. Al termine del tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni. Attraverso questa guida è possibile creare manualmente ogni risorsa richiesta per un cluster Amazon EKS. Le procedure offrono una visibilità completa sul modo in cui ogni risorsa viene creata e sul modo in cui interagiscono tra loro.

Offriamo anche i seguenti riferimenti:

- Per una raccolta di tutorial pratici, consulta EKS Cluster Setup su AWSCommunity.
- Per esempi di codice, consulta Code examples for Amazon EKS using AWS SDKs.

## Nozioni di base su Amazon EKS: EKS Auto Mode

Come altre esperienze introduttive con EKS, la creazione del primo cluster con EKS Auto Mode delega la gestione del cluster stesso a AWS. Tuttavia, EKS Auto Mode estende l'automazione EKS affidando la responsabilità di molti servizi essenziali necessari per configurare l'infrastruttura dei carichi di lavoro (nodi, reti e vari servizi), semplificando la gestione dei nodi e la scalabilità per soddisfare le richieste del carico di lavoro.

Scegli uno dei seguenti modi per creare un cluster con EKS Auto Mode:

- the section called " AWS CLI": utilizza l'interfaccia a riga di comando `aws` per creare un cluster.

- the section called "Console di gestione": utilizza Console di gestione AWS per creare un cluster.
- the section called "CLI di eksctl": utilizza l'interfaccia a riga di comando `eksctl` per creare un cluster.

Se stai confrontando diversi approcci alla creazione del vostro primo cluster EKS, devi sapere che EKS Auto Mode AWS si è assunta ulteriori responsabilità di gestione dei cluster, tra cui la configurazione di componenti per:

- Avvio e scalabilità dei nodi man mano che la domanda di carico di lavoro aumenta e diminuisce.
- Aggiornare regolarmente il cluster stesso (piano di controllo), i sistemi operativi dei nodi e i servizi in esecuzione sui nodi.
- Scelta delle impostazioni predefinite che determinano elementi come la dimensione e la velocità dell'archivio dei nodi e la configurazione della rete Pod.

Per informazioni sui vantaggi ottenibili con i cluster EKS Auto Mode, consulta Modalità automatica EKS.

## Inizia a usare Amazon EKS — **`eksctl`**

> **Note**
>
> Questo argomento illustra come iniziare senza modalità automatica di EKS.
> La modalità automatica di EKS automatizza le attività di routine per il cluster computing, l'archiviazione e la rete. Scopri come iniziare a usare Amazon EKS Auto Mode.

Questa guida consente di creare tutte le risorse necessarie per iniziare a usare Amazon Elastic Kubernetes Service (Amazon EKS) con `eksctl`, una semplice utility a riga di comando per la creazione e la gestione di cluster di Kubernetes su Amazon EKS. Al termine di questo tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni.

Le procedure descritte in questa guida consentono di creare automaticamente diverse risorse che è necessario creare manualmente quando si crea il cluster utilizzando il comando Console di gestione AWS. Se preferisci creare manualmente la maggior parte delle risorse per capire meglio come interagiscono tra loro, usa queste Console di gestione AWS per creare il cluster e il calcolo. Per ulteriori informazioni, consulta the section called "Crea cluster (console e CLI)".

## Prerequisiti

Prima di iniziare questo tutorial, devi installare e configurare gli strumenti AWS CLI, kubectl ed eksctl come descritto in Configurare l'uso di Amazon EKS.

## Passaggio 1: Creazione di cluster e nodi Amazon EKS

> **Important**
>
> Per iniziare nel modo più semplice e rapido possibile, in questo argomento sono inclusi i passaggi per creare un cluster e nodi con impostazioni predefinite. Prima di creare un cluster e i nodi da utilizzare in produzione, è consigliabile acquisire familiarità con tutte le impostazioni e implementare il cluster e nodi con le impostazioni che soddisfano i requisiti. Per ulteriori informazioni, consulta the section called "Creazione di un cluster " e Gestione del calcolo. Alcune impostazioni possono essere abilitate solo quando crei cluster e nodi.

È possibile creare un cluster con uno dei seguenti tipi di nodo. Per ulteriori informazioni sui tipi di nodi, consultare Gestione del calcolo. Dopo aver implementato il cluster, puoi aggiungere altri tipi di nodo.

- Fargate – Linux: selezionare questo tipo di nodo se si desidera eseguire applicazioni Linux su the section called " AWS Fargate". Fargate è un motore di elaborazione serverless che consente di implementare pod di Kubernetes senza gestire le istanze Amazon EC2.
- Nodi gestiti – Linux – Selezionare questo tipo di nodo se si desidera eseguire applicazioni Amazon Linux su istanze Amazon EC2. Sebbene non sia trattato in questa guida, è anche possibile aggiungere nodi autogestiti di Windows e Bottlerocket al cluster.

Creare il cluster Amazon EKS con il seguente comando. Sostituire *my-cluster* con il proprio valore. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e dell' AWS account in cui stai creando il cluster. Sostituiscilo *region-code* con qualsiasi AWS regione supportata da Amazon EKS. Per un elenco di AWS regioni, consulta gli endpoint e le quote di Amazon EKS nella guida di riferimento AWS generale.

Example

Fargate - Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes - Linux

```
eksctl create cluster --name my-cluster --region region-code
```

La creazione di cluster richiede diversi minuti. Durante la creazione, vengono generate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

`eksctl` ha creato un file di configurazione `kubectl` in `~/.kube/config` o ha aggiunto la configurazione del nuovo cluster all'interno di un file di configurazione esistente in `~/.kube/config` sul tuo computer.

Una volta completata la creazione del cluster, visualizza lo AWS CloudFormation stack indicato `eksctl-`*`my-cluster`*`-cluster` nella AWS CloudFormation console per vedere tutte le risorse che sono state create.

## Passaggio 2: Visualizzazione delle risorse Kubernetes

1. Visualizzare i nodi del cluster.

   ```
   kubectl get nodes -o wide
   ```

   Di seguito viene riportato un output di esempio.

   Example

   Fargate - Linux

   ```
   NAME                                                STATUS   ROLES    AGE
    VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
                   CONTAINER-RUNTIME
   ```

```
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
 8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
 7m30s   v1.2.3-eks-1234567   192.0.2.1     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes - Linux

```
NAME                                          STATUS   ROLES    AGE    VERSION
        INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
   CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
  v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
  v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Per ulteriori informazioni sugli elementi visualizzati nell'output, consultare the section called "Accesso alle risorse di cluster".

2. Visualizzazione dei carichi di lavoro in esecuzione nel cluster.

```
kubectl get pods -A -o wide
```

Di seguito viene riportato un output di esempio.

Example

Fargate - Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
          NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
 192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
 192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

Managed nodes - Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
          NODE                                                NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345                1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
kube-system   aws-node-67890                1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-abcde      1/1     Running   0          14m
 192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-12345      1/1     Running   0          14m
 192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-12345              1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-67890              1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
```

Per ulteriori informazioni sugli elementi visualizzati nell'output, consultare the section called "Accesso alle risorse di cluster".

## Passaggio 3: Eliminazione del cluster e dei nodi

Dopo aver creato il cluster e i nodi per questo tutorial, è consigliabile eseguire la pulizia mediante l'eliminazione del cluster e dei nodi con il seguente comando. Per eseguire altre operazioni con questo cluster prima di eseguire la pulizia, consultare the section called "Fasi successive".

```
eksctl delete cluster --name my-cluster --region region-code
```

## Fasi successive

I seguenti argomenti della documentazione consentono di estendere la funzionalità del cluster.

- Implementare un'applicazione di esempio al cluster.
- Il principale IAM in grado di creare il cluster è l'unico principale che può effettuare chiamate al server API Kubernetes tramite `kubectl` o la Console di gestione AWS. Se si desidera che altri principali IAM abbiano accesso al cluster, è necessario aggiungerli. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes" e the section called "Autorizzazioni richieste".

- Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di acquisire familiarità con tutte le impostazioni di cluster e nodi. Alcune impostazioni, come l'abilitazione dell'accesso SSH ai nodi Amazon EC2, devono essere definite al momento della creazione del cluster.
- Per aumentare la sicurezza del cluster, configura il plugin Amazon VPC Container Networking Interface per utilizzare i ruoli IAM per gli account di servizio.

# Inizia a usare Amazon EKS — Console di gestione AWS and AWS CLI

> **Note**
>
> Questo argomento illustra come iniziare senza modalità automatica di EKS. Utilizza i Gruppi di nodi gestiti per distribuire i nodi.
> La modalità automatica di EKS automatizza le attività di routine per il cluster computing, l'archiviazione e la rete. Scopri come iniziare a usare Amazon EKS Auto Mode. La modalità automatica di EKS è il metodo preferito per distribuire i nodi.

Questa guida ti aiuta a creare tutte le risorse necessarie per iniziare a usare Amazon Elastic Kubernetes Service (Amazon EKS) utilizzando Console di gestione AWS e la CLI. AWS Con questa guida è possibile creare manualmente ogni risorsa. Al termine di questo tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni.

Le procedure descritte in questa guida ti consentono di visualizzare in modo completo il modo in cui ogni risorsa viene creata e il modo in cui le risorse interagiscono tra loro. Se preferisci creare automaticamente la maggior parte delle risorse, utilizza la CLI `eksctl` per creare il cluster e i nodi. Per ulteriori informazioni, consulta the section called "Crea cluster (eksctl)".

## Prerequisiti

Prima di iniziare questo tutorial, è necessario installare e configurare i seguenti strumenti e risorse necessarie per creare e gestire un cluster Amazon EKS.

- AWS CLI: uno strumento a riga di comando per lavorare con i AWS servizi, incluso Amazon EKS. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando. Dopo aver installato la AWS CLI, ti consigliamo di configurarla anche tu. Per ulteriori informazioni, consulta Configurazione rapida con aws configure nella Guida per

l'utente dell'interfaccia a riga di AWS comando. Nota che la AWS CLI v2 è necessaria per utilizzare l'opzione update-kubeconfig mostrata in questa pagina.

- **`kubectl`**— Uno strumento a riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called “Configura kubectl ed eksctl”.
- Autorizzazioni IAM richieste: il principio di sicurezza IAM che stai utilizzando deve disporre delle autorizzazioni per funzionare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi, un VPC e le risorse correlate. AWS CloudFormation Per ulteriori informazioni, consulta le pagine Actions e Using service-linked roles nella Guida per l'utente IAM. È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:

```
aws sts get-caller-identity
```

Ti consigliamo di completare la procedura descritta in questo argomento in una shell Bash. In alternativa, puoi apportare alcune modifiche alla tua shell per alcuni comandi di script, come i caratteri di continuazione della riga, e per il modo in cui le variabili vengono impostate e utilizzate. Inoltre, le regole di escape e di utilizzo delle virgolette per la shell (interprete di comandi) potrebbero essere diverse. Per ulteriori informazioni, consulta Utilizzo delle virgolette con stringhe nella AWS CLI nella Guida per l'utente dell'interfaccia a riga di comando. AWS

## Passaggio 1: Creazione del cluster Amazon EKS;

> **Important**
>
> Per iniziare nel modo più semplice e rapido possibile, in questo argomento sono inclusi i passaggi per creare un cluster con impostazioni predefinite. Prima di creare un cluster da utilizzare in produzione, è consigliabile acquisire familiarità con tutte le impostazioni e implementare un cluster con le impostazioni che soddisfano i tuoi requisiti. Per ulteriori informazioni, consulta the section called “Creazione di un cluster ”. Alcune impostazioni possono essere abilitate solo quando crei cluster.

1. Crea un Amazon VPC con sottoreti pubbliche e private che soddisfino i requisiti Amazon EKS. Sostituisci *`region-code`* con qualsiasi AWS regione supportata da Amazon EKS. Per un elenco di AWS regioni, consulta gli endpoint e le quote di Amazon EKS nella guida di riferimento AWS generale. Puoi sostituire *`my-eks-vpc-stack`* con un nome a tua scelta.

```
aws cloudformation create-stack \
  --region region-code \
  --stack-name my-eks-vpc-stack \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

> **Tip**
>
> Per un elenco di tutte le risorse create dal comando precedente, apri la AWS CloudFormation console in https://console.aws.amazon.com/cloudformation/. Scegli *my-eks-vpc-stack* lo stack e seleziona la scheda  Risorse.

2. Crea un ruolo IAM del cluster e collegalo alla policy gestita IAM richiesta dall'AMI per Amazon EKS. I cluster Kubernetes gestiti da Amazon EKS effettuano chiamate ad altri AWS servizi per tuo conto per gestire le risorse che utilizzi con il servizio.

   a. Copiare i seguenti contenuti in un file denominato *eks-cluster-role-trust-policy.json*.

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "eks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. Creare il ruolo.

   ```
   aws iam create-role \
     --role-name myAmazonEKSClusterRole \
     --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
   ```

   c. Allega la policy IAM gestita da Amazon EKS richiesta al ruolo.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name myAmazonEKSClusterRole
```

3. Apri la console Amazon EKS all'indirizzo #/clusters. https://console.aws.amazon.com/eks/home

   Assicurati che la AWS regione mostrata in alto a destra della console sia la AWS regione in cui desideri creare il cluster. In caso contrario, scegli il menu a discesa accanto al nome AWS della regione e scegli la AWS regione che desideri utilizzare.

4. Scegli Crea cluster. Se non viene visualizzata questa opzione, scegli Cluster nel pannello di navigazione a sinistra.

5. Nella pagina Configura cluster, completa le seguenti operazioni:

   a. Seleziona Configurazione personalizzata e disabilita Usa modalità automatica di EKS (se preferisci un cluster della modalità automatica di EKS, consulta invece the section called "Console di gestione").

   b. Immettere un nome per il cluster, ad esempio *my-cluster*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.

   c. Per Cluster Service Role, scegli*myAmazonEKSClusterRole*.

   d. Lascia le impostazioni rimanenti sui valori di default e seleziona Successivo.

6. Nella pagina Specifica rete, procedi come segue:

   a. Scegli l'ID del VPC creato in un passaggio precedente nell'elenco a discesa VPC. È qualcosa del genere** | my-eks-vpc-stack-VPC*.

   b. Seleziona le sottoreti create in un passaggio precedente nell'elenco a discesa Sottoreti. Le sottoreti saranno qualcosa di simile. ** | my-eks-vpc-stack-**

   c. Seleziona il gruppo di sicurezza creato in un passaggio precedente nell'elenco a discesa Gruppi di sicurezza aggiuntivi. È qualcosa di simile. ** | my-eks-vpc-stack-ControlPlaneSecurityGroup-**

   d. Lascia le impostazioni rimanenti sui valori di default e seleziona Successivo.

7. Nella pagina Configura osservabilità, scegli Avanti.

8. Nella pagina Seleziona componenti aggiuntivi, scegli Successivo.

Per ulteriori informazioni sui componenti aggiuntivi, consulta la pagina the section called "Componenti aggiuntivi Amazon EKS".

9. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, scegli Successivo.
10. Nella pagina Rivedi e crea, scegli Crea.

    A destra del nome del cluster, lo stato del cluster è Creazione in corso per alcuni minuti fino al termine del processo di provisioning del cluster. Non proseguire con il passaggio successivo finché lo stato non è Attivo.

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

## Fase 2: Connettiti al tuo cluster

In questa sezione, ti connetti `kubectl` al tuo cluster in modo da poter gestire le risorse Kubernetes.

Prima di procedere, assicurati che la creazione del cluster sia stata completata correttamente nel passaggio 1.

### Opzione 1: connettiti utilizzando AWS CLI

1. Crea o aggiorna un file `kubeconfig` per il cluster. Sostituisci *`region-code`* con la AWS regione in cui hai creato il cluster. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

   Per impostazione predefinita, il file `config` viene creato in `~/.kube` o la configurazione del nuovo cluster viene aggiunta a un file `config` esistente in `~/.kube`.
2. Prova la configurazione.

```
kubectl get svc
```

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso non autorizzato o negato (kubectl) " nell'argomento relativo alla risoluzione dei problemi.

Di seguito viene riportato un output di esempio.

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

### Opzione 2: connettiti usando la console EKS

In alternativa, puoi connetterti direttamente dalla console senza alcuna configurazione locale:

1. Nella pagina dei dettagli del cluster per il cluster creato nel passaggio 1, scegli Connetti.
2. AWS CloudShell si apre con una `kubectl` configurazione preconfigurata per il tuo cluster.
3. Verifica la tua connessione:

```
kubectl get svc
```

Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".

## Passaggio 3: Creazione di nodi

> **Important**
>
> Per iniziare nel modo più semplice e rapido possibile, in questo argomento sono inclusi i passaggi per creare nodi con la maggior parte delle impostazioni predefinite. Prima di creare i nodi da utilizzare in produzione, è consigliabile acquisire familiarità con tutte le impostazioni e implementare i nodi con le impostazioni che soddisfano i tuoi requisiti. Per

ulteriori informazioni, consulta Gestione del calcolo. Alcune impostazioni possono essere abilitate solo quando crei i nodi.

Questa procedura configura il cluster per utilizzare gruppi di nodi gestiti per creare nodi, specificando le sottoreti e il ruolo IAM del nodo creato nei passaggi precedenti. Consente di eseguire applicazioni Amazon Linux su istanze Amazon EC2.

Per ulteriori informazioni sui diversi modi di configurare i nodi in EKS, consulta Gestione del calcolo. Dopo aver implementato il cluster, puoi aggiungere altri tipi di nodo. Sebbene non sia trattato in questa guida, è anche possibile aggiungere nodi autogestiti di Windows e Bottlerocket al cluster.

Per creare un gruppo di nodi gestiti Linux di Amazon EC2

1. Creare un ruolo IAM del nodo e allegarvi la policy gestita IAM di Amazon EKS richiesta. Il `kubelet` daemon del nodo Amazon EKS effettua chiamate alle AWS API per tuo conto. I nodi ricevono le autorizzazioni per queste chiamate API attraverso un profilo dell'istanza IAM e le policy associate.
   a. Copia i seguenti contenuti in un file denominato `node-role-trust-policy.json`.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   b. Crea il ruolo IAM del nodo.

```
aws iam create-role \
  --role-name myAmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-policy.json"
```

   c. Allega al ruolo le policy gestite IAM richieste.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

d. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.

e. Scegli il nome del cluster che hai creato nella Fase 1: Crea il tuo cluster Amazon EKS, ad esempio. *my-cluster*

f. Nella ***my-cluster*** pagina, procedi come segue:

g. Scegli la scheda Calcolo.

h. Scegli Aggiungi gruppo di nodi.

2. Nella pagina Configure Node Group (Configura gruppo di nodi), effettuare le seguenti operazioni:

   a. Per Nome, immettere un nome univoco per il gruppo di nodi gestiti, ad esempio *my-nodegroup*. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e trattini bassi per i caratteri rimanenti.

   b. Per il nome del ruolo IAM di Node, scegli il *myAmazonEKSNodeRole* ruolo che hai creato in un passaggio precedente. Consigliamo che ciascun gruppo di nodi utilizzi il proprio ruolo IAM univoco.

   c. Scegli Avanti.

3. Sulla pagina Set compute and scaling configuration (Impostazione della configurazione di calcolo e dimensionamento) accettare i valori di default e selezionare Next (Successivo).

4. Nella pagina Specify networking (Specifica rete), accettare i valori di default e selezionare Next (Successivo).

5. Nella pagina Rivedi e crea, controlla la configurazione del gruppo di nodi gestiti e scegli Crea.

6. Dopo alcuni minuti, lo Stato nella Configurazione del gruppo di nodi cambierà da In Passaggio di creazione a Attivo. Non proseguire con il passaggio successivo finché lo stato non è Attivo.

## Passaggio 4: Visualizzazione delle risorse

È possibile visualizzare i nodi e i carichi di lavoro Kubernetes.

1. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli il nome del cluster che hai creato, ad esempio*my-cluster*.
2. Nella ***my-cluster*** pagina, scegli quanto segue:
   a. Scheda Calcolo: verrà visualizzato l'elenco di nodi che sono stati implementati per il cluster. È possibile scegliere il nome di un nodo per visualizzare ulteriori informazioni su di esso.
   b. Scheda Risorse: verranno visualizzate tutte le risorse Kubernetes che sono state implementate per impostazione predefinita in un cluster Amazon EKS. Seleziona qualsiasi tipo di risorsa nella console per avere ulteriori informazioni in merito.

## Passaggio 5. Eliminazione delle risorse

Dopo aver creato il cluster e i nodi per questo tutorial, devi eliminare le risorse create. Per eseguire altre operazioni con questo cluster prima di eliminare le risorse, consultare the section called "Fasi successive".

1. Eliminare qualsiasi gruppo di nodi o i profili Fargate creati.
   a. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
   b. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli *my-cluster*.
   c. Scegli la scheda Calcolo.
   d. Se hai creato un gruppo di nodi, scegli il gruppo di *my-nodegroup* nodi e quindi scegli Elimina. Immettere *my-nodegroup*, quindi scegliere Delete (Elimina).
   e. Non continuare fino a quando il gruppo di nodi o i profili non saranno stati eliminati.
2. Elimina il cluster.
   a. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli *my-cluster*.
   b. Scegli Delete cluster (Elimina cluster).
   c. Immetti *my-cluster*, quindi scegli Elimina. Non continuare fino a quando il cluster non sarà stato eliminato.
3. Elimina lo AWS CloudFormation stack VPC che hai creato.
   a. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation/.
   b. Scegli lo stack *my-eks-vpc-stack*, quindi seleziona Delete (Elimina).

c. Nella finestra di dialogo di ***my-eks-vpc-stack*** conferma dell'eliminazione, scegli Elimina stack.

4. Elimina i ruoli IAM creati.
   a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
   b. Nel pannello di navigazione a sinistra, seleziona Ruoli.
   c. Seleziona ogni ruolo che hai creato dall'elenco (***myAmazonEKSClusterRole***e anche*myAmazonEKSNodeRole*). Scegli Elimina, inserisci il testo di conferma richiesto, quindi scegli Elimina.

## Fasi successive

I seguenti argomenti della documentazione consentono di estendere la funzionalità del cluster.

- Il principale IAM in grado di creare il cluster è l'unico principale che può effettuare chiamate al server API Kubernetes tramite `kubectl` o la Console di gestione AWS. Se si desidera che altri principali IAM abbiano accesso al cluster, è necessario aggiungerli. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes" e the section called "Autorizzazioni richieste".
- Implementare un'applicazione di esempio al cluster.
- Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di acquisire familiarità con tutte le impostazioni di cluster e nodi. Alcune impostazioni, come l'abilitazione dell'accesso SSH ai nodi Amazon EC2, devono essere definite al momento della creazione del cluster.
- Per aumentare la sicurezza del cluster, configura il plugin Amazon VPC Container Networking Interface per utilizzare i ruoli IAM per gli account di servizio.

# Automate cluster infrastructure with EKS Auto Mode

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS Auto Mode.

EKS Auto Mode estende la AWS gestione dei cluster Kubernetes oltre il cluster stesso, per consentire AWS anche di configurare e gestire l'infrastruttura che consente il corretto funzionamento dei carichi di lavoro. Puoi delegare le decisioni chiave sull'infrastruttura e utilizzare l'esperienza di per le operazioni quotidiane. AWS L'infrastruttura cluster gestita da AWS include molte funzionalità di Kubernetes come componenti principali, al contrario di componenti aggiuntivi, come scalabilità automatica di calcolo, rete di pod e servizi, bilanciamento del carico delle applicazioni, DNS del cluster, storage a blocchi e supporto GPU.

Per iniziare, è possibile implementare un nuovo cluster con modalità automatica EKS o abilitare la modalità automatica EKS su un cluster esistente. Puoi distribuire, aggiornare o modificare i tuoi cluster EKS Auto Mode utilizzando eksctl, la CLI, le API EKS o i tuoi strumenti infrastructure-as-code preferiti. AWS Console di gestione AWS

Con EKS Auto Mode, puoi continuare a utilizzare i tuoi strumenti preferiti. Kubernetes-compatible EKS Auto Mode si integra con AWS servizi come Amazon EC2, Amazon EBS ed ELB, utilizzando risorse AWS cloud che seguono le best practice. Queste risorse vengono dimensionate automaticamente, ottimizzate in termini di costi e aggiornate regolarmente per contribuire a ridurre al minimo i costi operativi e le spese generali.

## Funzionalità

modalità automatica di EKS offre le seguenti funzionalità di alto livello:

Ottimizzazione della gestione dei cluster Kubernetes: la modalità automatica EKS semplifica la gestione di EKS fornendo cluster pronti per la produzione con un sovraccarico operativo minimo. Con la modalità automatica EKS, puoi eseguire carichi di lavoro impegnativi e dinamici in sicurezza, senza bisogno di una profonda conoscenza di EKS.

Disponibilità delle applicazioni: modalità automatica di EKS aggiunge o rimuove dinamicamente i nodi nel cluster EKS in base alle esigenze delle applicazioni Kubernetes. Questo riduce al minimo la

necessità di pianificare manualmente la capacità e garantisce la disponibilità delle applicazioni. EKS Auto Mode si integra anche con lo spostamento zonale di Amazon Application Recovery Controller (ARC) per interrompere automaticamente il provisioning dei nodi in una zona di disponibilità compromessa e sospendere le interruzioni volontarie durante un evento di spostamento zonale.

Efficienza: EKS Auto Mode è progettata per ottimizzare i costi di elaborazione rispettando al contempo la flessibilità definita dai requisiti dell'utente e del carico di lavoro. NodePool Inoltre, termina le istanze non utilizzate e consolida i carichi di lavoro su altri nodi per migliorare l'efficienza in termini di costi. Per indicazioni sulla configurazione del cluster per ridurre i costi ed evitare configurazioni che bloccano il consolidamento, vedi Ottimizzazione dei costi in EKS Auto Mode.

Sicurezza: EKS Auto Mode utilizza AMI considerate immutabili per i nodi. Tali AMI applicano software bloccato, abilitano i controlli di accesso obbligatori di SELinux e forniscono file system root di sola lettura. Inoltre, i nodi avviati dalla modalità automatica EKS hanno una durata massima di 21 giorni (che può essere ridotta), dopodiché vengono sostituiti automaticamente con nuovi nodi. Questo approccio migliora il livello di sicurezza grazie al riciclo regolare dei nodi, in linea con le best practice già adottate da molti clienti.

Aggiornamenti automatici: EKS Auto Mode mantiene il cluster Kubernetes, i nodi e i relativi componenti aggiornati con le ultime patch, rispettando al contempo i Pod Disruption Budget (PDB) e i Disruption Budget (NDB) configurati. NodePool Fino al limite massimo di 21 giorni, potrebbe essere necessario un intervento se il blocco dei PDB o altre configurazioni impediscono gli aggiornamenti. Per il rollback delle versioni, EKS Auto Mode ripristina automaticamente i nodi di lavoro prima che il piano di controllo venga ripristinato, rispettando i controlli di interruzione configurati. Per ulteriori informazioni, consulta the section called "Ripristina i cluster EKS Auto Mode".

Componenti gestiti: EKS Auto Mode include Kubernetes e funzionalità AWS cloud come componenti principali che altrimenti dovrebbero essere gestiti come componenti aggiuntivi. Sono inclusi il supporto integrato per l'assegnazione degli indirizzi IP dei pod, le policy di rete dei pod, i servizi DNS locali, i componenti aggiuntivi GPU, i controlli di integrità e lo storage EBS CSI.

Personalizzabile NodePools e NodeClasses: se il carico di lavoro richiede modifiche alle configurazioni di archiviazione, elaborazione o rete, è possibile creare configurazioni personalizzate utilizzando la modalità EKS Auto. NodePools NodeClasses Anche se non è necessario modificare le impostazioni predefinite NodePools NodeClasses, è possibile aggiungere nuove configurazioni personalizzate NodePools o NodeClasses parallele a quelle predefinite per soddisfare i requisiti specifici.

# Componenti automatici

La modalità automatica EKS semplifica il funzionamento dei cluster Amazon EKS automatizzando i componenti chiave dell'infrastruttura. L'attivazione della modalità automatica EKS riduce ulteriormente le attività di gestione dei cluster EKS.

Di seguito è riportata una lista di componenti del piano dati che sono automatizzati:

- Calcolo: per molti carichi di lavoro, con la modalità automatica EKS puoi dimenticarti di molti aspetti relativi al calcolo per i cluster EKS. Ciò include:
  - Nodi: i nodi della modalità automatica EKS sono progettati per essere trattati come dispositivi. La modalità automatica EKS esegue le seguenti operazioni:
    - Sceglie un'AMI appropriata configurata con molti servizi necessari per eseguire i carichi di lavoro senza bisogno di intervento.
    - Blocca l'accesso ai file sull'AMI usando la modalità di applicazione SELinux e un file system root di sola lettura.
    - Impedisce l'accesso diretto ai nodi disabilitando l'accesso SSH o SSM.
    - Include il supporto GPU, con driver del kernel e componenti aggiuntivi separati per GPU NVIDIA e Neuron, che consente carichi di lavoro con prestazioni elevate.
    - Gestisce automaticamente gli avvisi di interruzione delle istanze Spot EC2, gli eventi di health/scheduled manutenzione delle istanze EC2 e gli errori di controllo dello stato delle istanze EC2
  - Scalabilità automatica: basandosi sul ridimensionamento automatico di Karpenter, EKS Auto Mode monitora i pod non programmabili e rende possibile l'implementazione di nuovi nodi per eseguire tali pod. Quando i carichi di lavoro vengono terminati, la modalità automatica EKS interrompe e termina in modo dinamico i nodi quando non sono più necessari, ottimizzando l'uso delle risorse.
  - Aggiornamenti: il controllo dei nodi semplifica la capacità della modalità automatica EKS di offrire patch di sicurezza e aggiornamenti del sistema operativo e dei componenti in base alle esigenze. Tali aggiornamenti sono progettati per ridurre al minimo le interruzioni dei carichi di lavoro. La modalità automatica di EKS impone una durata massima dei nodi di 21 giorni per garantire che il software e le API siano sempre aggiornati.
- Bilanciamento del carico: modalità automatica di EKS semplifica il bilanciamento del carico integrandosi con il servizio Elastic Load Balancing di Amazon, automatizzando il provisioning e la configurazione dei bilanciatori di carico per i servizi Kubernetes e le risorse Ingress. Supporta funzionalità avanzate per Application e Network Load Balancer, ne gestisce il ciclo di vita e le

dimensiona in base alle esigenze del cluster. Questa integrazione fornisce una soluzione di bilanciamento del carico pronta per la produzione che aderisce alle best practice, consentendoti di concentrarti sulle applicazioni anziché sulla gestione dell'infrastruttura AWS .

- Storage: modalità automatica di EKS configura automaticamente lo storage temporaneo impostando i tipi di volume, le dimensioni dei volumi, le policy di crittografia e le policy di eliminazione alla chiusura del nodo.
- Rete: modalità automatica di EKS automatizza le attività di rete critiche per quanto riguarda la connettività dei pod e dei servizi. Ciò include il IPv4/IPv6 supporto e l'uso di blocchi CIDR secondari per estendere gli spazi degli indirizzi IP.
- Identity and Access Management: non è necessario installare l'agente EKS Pod Identity sui cluster in modalità automatica di EKS.

Per ulteriori informazioni su questi componenti, consulta the section called "Come funziona".

## Configurazione

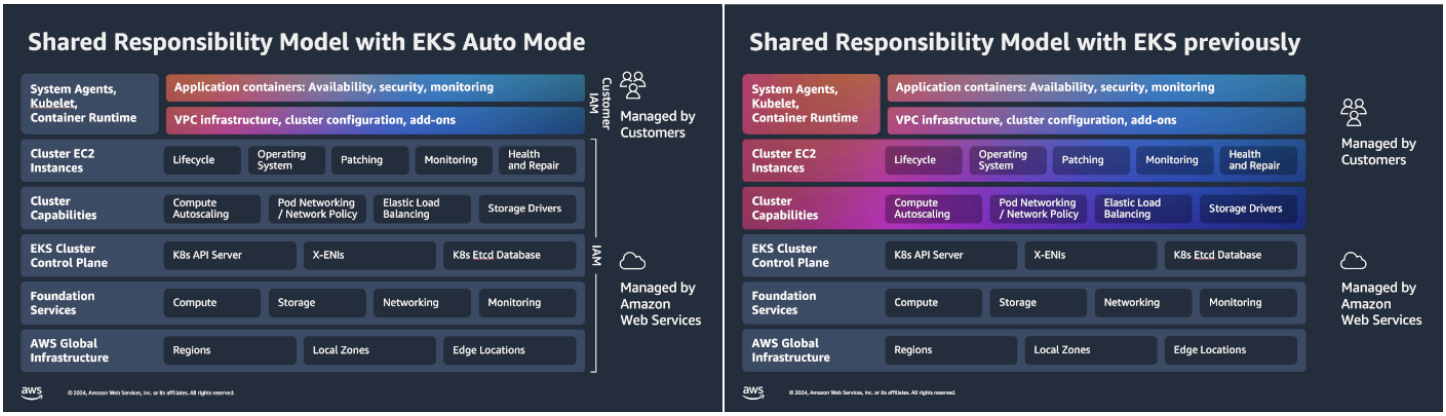
Sebbene modalità automatica di EKS gestisca in modo efficace la maggior parte dei servizi del piano dati senza alcun intervento da parte tua, potrebbero esserci casi in cui desideri modificare il comportamento di alcuni di questi servizi. Puoi modificare la configurazione dei cluster della modalità automatica di EKS nei seguenti modi:

- Kubernetes DaemonSets: anziché modificare i servizi installati sui nodi, puoi utilizzare Kubernetes. DaemonSets Kubernetes gestisce, che vengono eseguiti su tutti i nodi del DaemonSets cluster. In questo modo, puoi aggiungere servizi speciali per il monitoraggio o la sorveglianza dei nodi.
- Personalizzati NodePools e NodeClasses: predefiniti NodePools e NodeClasses sono configurati da EKS Auto Mode e non è necessario modificarli. Per personalizzare il comportamento dei nodi, è possibile creare casi d'uso aggiuntivi NodePools o NodeClasses per casi d'uso come:
  - La selezione di tipi di istanze specifici (ad esempio, processori accelerati o istanze Spot EC2).
  - L'isolamento dei carichi di lavoro per scopi di sicurezza o di monitoraggio dei costi.
  - La configurazione di impostazioni di storage temporaneo come IOPS, dimensioni e throughput.
- Bilanciamento del carico: alcuni servizi, come il bilanciamento del carico, che modalità automatica di EKS esegue come oggetti Kubernetes, possono essere configurati direttamente sui cluster della modalità automatica di EKS.

Per ulteriori informazioni sulle opzioni di configurazione della modalità automatica di EKS, consulta the section called "Configura".

# Modello di responsabilità condivisa

Il modello di responsabilità AWS condivisa definisce le responsabilità di sicurezza e conformità tra AWS e i clienti. Le immagini e il testo seguenti confrontano e contrappongono AWS le differenze tra clienti e responsabilità tra la modalità EKS Auto e la modalità standard EKS.




La modalità automatica di EKS trasferisce gran parte della responsabilità condivisa per l'infrastruttura Kubernetes dai clienti ad AWS. Con EKS Auto Mode, AWS si assume una maggiore responsabilità per la sicurezza del cloud, che una volta era responsabilità del cliente e ora è condivisa. I clienti possono ora concentrarsi maggiormente sulle proprie applicazioni AWS gestendo al contempo l'infrastruttura sottostante.

Responsabilità del cliente

Con la modalità automatica di EKS, i clienti continuano a mantenere la responsabilità dei container delle applicazioni, inclusi disponibilità, sicurezza e monitoraggio. Mantengono inoltre il controllo sull'infrastruttura VPC e sulla configurazione del cluster EKS. Questo modello consente ai clienti di concentrarsi sui problemi specifici delle applicazioni, delegando al contempo la gestione dell'infrastruttura del cluster a. AWS Le funzionalità opzionali per nodo possono essere incluse nei cluster tramite componenti aggiuntivi. AWS

AWS responsabilità

Con EKS Auto Mode, AWS amplia la propria responsabilità includendo la gestione di diversi componenti critici aggiuntivi rispetto a quelli già gestiti nei cluster EKS che non utilizzano la modalità automatica. In particolare, modalità automatica di EKS si occupa della configurazione, della gestione,

della sicurezza e del dimensionamento delle istanze EC2 avviate, nonché delle funzionalità del cluster per il bilanciamento del carico, la gestione degli indirizzi IP, la policy di rete e lo storage a blocchi. I seguenti componenti sono gestiti da AWS EKS Auto Mode:

- Istanze Mode-launched EC2 automatiche: AWS gestisce l'intero ciclo di vita dei nodi utilizzando istanze gestite da Amazon EC2. Le istanze gestite da EC2 si assumono la responsabilità della configurazione del sistema operativo, dell'applicazione di patch, del monitoraggio e del mantenimento dell'integrità. In questo modello, sia l'istanza stessa che il sistema operativo guest in esecuzione su di essa sono di responsabilità di AWS. I nodi usano varianti delle AMI Bottlerocket ottimizzate per l'esecuzione di container. Le AMI Bottlerocket dispongono di software bloccato, file system root immutabili e accesso alla rete sicuro (per impedire comunicazioni dirette tramite SSH o SSM). A partire dal 22 aprile 2026, le nuove istanze gestite e le risorse associate sono nascoste per impostazione predefinita dalle visualizzazioni della console EC2 e dalle operazioni degli elenchi di API. Puoi modificare questo comportamento con le impostazioni di visibilità delle risorse gestite. Per ulteriori informazioni, consulta the section called "Istanze gestite".
- Funzionalità del cluster: AWS gestisce la scalabilità automatica dell'elaborazione, il networking Pod con l'applicazione delle policy di rete, l'integrazione di Elastic Load Balancing e la configurazione dei driver di archiviazione.
- Cluster Control Plane: AWS continua a gestire il server API Kubernetes, gli ENI multiaccount e il database etcd, come nel caso dello standard EKS.
- Foundation Services and Global Infrastructure: AWS mantiene la responsabilità dei servizi di elaborazione, archiviazione, rete e monitoraggio sottostanti, nonché dell'infrastruttura globale di regioni, zone locali e sedi periferiche.

# Creare un cluster con la modalità automatica Amazon EKS

Questo capitolo spiega come creare un cluster Amazon EKS con la modalità automatica abilitata utilizzando vari strumenti e interfacce. La modalità automatica semplifica la creazione di cluster grazie alla configurazione e alla gestione automatiche dell'infrastruttura di calcolo, della rete e dell'archiviazione del cluster. Si imparerà come creare un cluster con la modalità automatica utilizzando la AWS CLI, la Console di gestione AWS o lo strumento a riga di comando eksctl.

> **Note**
>
> La modalità automatica EKS richiede la versione 1.29 o successive di Kubernetes.

Scegliere lo strumento preferito in base alle esigenze: la Console di gestione AWS offre un'interfaccia visiva ideale per scoprire le funzionalità della modalità automatica EKS e creare singoli cluster. La AWS CLI è la soluzione ideale per attività di scripting e automazione, in particolare quando si integra la creazione di cluster in flussi di lavoro o pipeline CI/CD esistenti. Lo strumento CLI eksctl offre un'esperienza nativa per Kubernetes ed è consigliato agli utenti che hanno familiarità con gli strumenti Kubernetes e che desiderano operazioni a riga di comando semplificate con impostazioni predefinite ragionevoli.

Prima di iniziare, assicurarsi di aver installato e configurato i prerequisiti necessari, incluse le autorizzazioni IAM adeguate per creare cluster EKS. Per ulteriori informazioni su come installare strumenti CLI come `kubectl`, `aws` e `eksctl`, consultare Configurazione.

È possibile utilizzare la AWS CLI, la Console di gestione AWS o lo strumento CLI eksctl per creare un cluster con la modalità automatica Amazon EKS.

Argomenti

- Crea un cluster della modalità automatica di EKS con la CLI di eksctl
- Crea un cluster EKS Auto Mode con AWS CLI
- Crea un cluster EKS Auto Mode con Console di gestione AWS

## Crea un cluster della modalità automatica di EKS con la CLI di eksctl

Questo argomento ti mostra come creare un cluster della modalità automatica di Amazon EKS con l'interfaccia a riga di comando (CLI) di eksctl. Puoi creare un cluster Auto Mode eseguendo un singolo comando CLI o applicando un file di configurazione YAML. Entrambi i metodi offrono le stesse funzionalità, con l'approccio YAML che offre un controllo più granulare sulle impostazioni del cluster.

La CLI eksctl semplifica il processo di creazione e gestione dei cluster EKS Auto Mode gestendo la creazione e la configurazione delle risorse sottostanti. AWS Prima di procedere, assicurati di avere le AWS credenziali e le autorizzazioni necessarie configurate sul tuo computer locale. Questa guida presuppone che tu abbia una certa familiarità con i concetti base di Amazon EKS e che gli strumenti CLI richiesti siano già stati installati.

> **Note**
>
> Devi installare la versione `0.195.0` o una versione successiva di eksctl. Per ulteriori informazioni, vedere eksctl releases on. GitHub

## Crea un cluster della modalità automatica di EKS con un comando CLI

Devi avere l'installazione degli strumenti `aws` ed `eksctl`. Devi accedere alla AWS CLI con autorizzazioni sufficienti per AWS gestire risorse tra cui: istanze EC2, reti EC2, cluster EKS e ruoli IAM. Per ulteriori informazioni, consulta Configurazione.

Esegui il seguente comando per creare un nuovo cluster EKS Auto Mode con eksctl:

```
eksctl create cluster --name=<cluster-name> --enable-auto-mode
```

## Crea un cluster della modalità automatica di EKS con un file YAML

Devi avere l'installazione degli strumenti `aws` ed `eksctl`. Devi accedere alla AWS CLI con autorizzazioni sufficienti per AWS gestire risorse tra cui: istanze EC2, reti EC2, cluster EKS e ruoli IAM. Per ulteriori informazioni, consulta Configurazione.

Esamina le opzioni di configurazione EKS Auto Mode nella risorsa di esempio riportata di seguito. ClusterConfig Per le ClusterConfig specifiche complete, consulta la documentazione di eksctl.

AWS suggerisce di abilitare la modalità automatica EKS. Se è la prima volta che crei un cluster della modalità automatica di EKS, lascia il `nodeRoleARN` non precisato per creare un ruolo IAM del nodo per la modalità automatica di EKS. Se hai già un ruolo Node IAM nel tuo AWS account, ti AWS suggerisce di riutilizzarlo.

AWS suggerisce di non specificare alcun valore per. `nodePools` modalità automatica di EKS creerà pool di nodi predefiniti. Puoi utilizzare l'API Kubernetes per creare pool di nodi aggiuntivi.

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: <cluster-name>
  region: <aws-region>
```

```
iam:
  # ARN of the Cluster IAM Role
  # optional, eksctl creates a new role if not supplied
  # suggested to use one Cluster IAM Role per account
  serviceRoleARN: <arn-cluster-iam-role>

autoModeConfig:
  # defaults to false
  enabled: boolean
  # optional, defaults to [general-purpose, system].
  # suggested to leave unspecified
  # To disable creation of nodePools, set it to the empty array ([]).
  nodePools: []string
  # optional, eksctl creates a new role if this is not supplied
  # and nodePools are present.
  nodeRoleARN: string
```

Salva il file `ClusterConfig` con nome `cluster.yaml` e usa il seguente comando per creare il cluster:

```
eksctl create cluster -f cluster.yaml
```

## Crea un cluster EKS Auto Mode con AWS CLI

I cluster della modalità automatica di EKS automatizzano le attività di routine di gestione del cluster l'elaborazione, l'archiviazione e la rete. Ad esempio, i cluster della modalità automatica di EKS rilevano automaticamente quando sono necessari nodi aggiuntivi ed effettuano il provisioning di nuove istanze EC2 per soddisfare le richieste di carico di lavoro.

Questo argomento guida l'utente nella creazione di un nuovo cluster EKS Auto Mode utilizzando la AWS CLI e, facoltativamente, nella distribuzione di un carico di lavoro di esempio.

### Prerequisiti

- L'ultima versione della AWS Command Line Interface (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.
  - Accedi alla CLI con autorizzazioni IAM sufficienti per creare AWS risorse tra cui politiche IAM, ruoli IAM e cluster EKS.

- Lo strumento da riga di comando kubectl installato sul tuo dispositivo. AWS suggerisce di utilizzare la stessa versione di kubectl della versione Kubernetes del cluster EKS. Per installare o aggiornare kubectl, consulta the section called "Configura kubectl ed eksctl".

## Specifica le sottoreti VPC

Amazon EKS Auto Mode distribuisce nodi su sottoreti VPC. Quando si crea un cluster EKS, devi specificare le sottoreti VPC in cui verranno implementati i nodi. Puoi utilizzare le sottoreti VPC predefinite nel tuo AWS account o creare un VPC dedicato per carichi di lavoro critici.

- AWS suggerisce di creare un VPC dedicato per il cluster. Informazioni su come the section called "Crea un VPC".
- La console EKS aiuta a creare un nuovo VPC. Informazioni su come the section called "Console di gestione".
- In alternativa, puoi utilizzare il VPC predefinito del tuo AWS account. Utilizza le seguenti istruzioni per trovare gli ID delle sottoreti.

Per trovare gli ID delle sottoreti del VPC predefinito

Utilizzando la AWS CLI:

1. Esegui il comando seguente per elencare il VPC di default e le relative sottoreti:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$(aws ec2 describe-vpcs
 --query 'Vpcs[?IsDefault==`true`].VpcId' --output text)" --query 'Subnets[*].
{ID:SubnetId,AZ:AvailabilityZone}' --output table
```

2. Salva l'output e annotare gli ID delle sottoreti.

   Output di esempio:

```
-----------------------------------------
|             DescribeSubnets           |
-----------------------------------------
|   SubnetId        |   AvailabilityZone  |
|-------------------|---------------------|
|   subnet-012345678 |   us-west-2a        |
|   subnet-234567890 |   us-west-2b        |
|   subnet-345678901 |   us-west-2c        |
```

```
---------------------------------------
```

## Ruoli IAM per cluster della modalità automatica di EKS

### Ruolo IAM del cluster

EKS Auto Mode richiede un ruolo Cluster IAM per eseguire azioni nel tuo AWS account, come il provisioning di nuove istanze EC2. È necessario creare questo ruolo per concedere a EKS le autorizzazioni necessarie. AWS consiglia di allegare le seguenti politiche AWS gestite al ruolo IAM del Cluster:

- AmazonEKSComputePolicy
- AmazonEKSBlockStoragePolicyV2
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy
- AmazonEKSClusterPolicy

### Ruolo IAM del nodo

Quando creai un cluster della modalità automatica di EKS, specifichi un ruolo IAM del nodo. Quando modalità automatica di EKS crea nodi per elaborare carichi di lavoro in sospeso, a ogni nuovo nodo di istanza EC2 viene assegnato il ruolo IAM del nodo. Questo ruolo consente al nodo di comunicare con EKS, ma in genere non è accessibile ai carichi di lavoro in esecuzione sul nodo.

Se desideri concedere le autorizzazioni ai carichi di lavoro in esecuzione su un nodo, usa EKS Pod Identity. Per ulteriori informazioni, consulta the section called "Pod Identity".

È necessario creare questo ruolo e allegare la seguente policy AWS gestita:

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

## Service-Linked Ruolo

La modalità automatica EKS richiede anche un Service-Linked ruolo, che viene creato e configurato automaticamente da AWS. Per ulteriori informazioni, consulta AWSServiceRoleForAmazonEKS.

## Creazione di un ruolo IAM per un cluster della modalità automatica di EKS

### Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

### Passaggio 2: creazione del ruolo IAM

Usa la policy di attendibilità per creare il ruolo IAM del cluster:

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

### Passaggio 3: annotazione del ruolo ARN

Recupera e salva l'ARN del nuovo ruolo per usarlo nelle fasi successive:

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

### Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti politiche AWS gestite al ruolo IAM del Cluster per concedere le autorizzazioni necessarie:

AmazonEKSClusterPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicyV2:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicyV2
```

AmazonEKSLoadBalancingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

## Creazione di un ruolo IAM per un nodo modalità automatica di EKS

Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `node-trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Passaggio 2: creare il ruolo IAM del nodo

Usa il file node-trust-policy.json del passaggio precedente per definire quali entità possono assumere il ruolo. Per creare il ruolo IAM del nodo, esegui il seguente comando:

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Passaggio 3: annotazione del ruolo ARN

Una volta creato il ruolo, recupera e salva l'ARN del ruolo IAM del nodo. Sarà necessario usare questo ARN nei passaggi successivi. Usa il seguente comando per ottenere l'ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti politiche AWS gestite al ruolo IAM del nodo per fornire le autorizzazioni necessarie:

AmazonEKSWorkerNodeMinimalPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

AmazonEC2ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

# Creazione di un cluster della modalità automatica di EKS

## Panoramica di

Per creare un cluster EKS Auto Mode utilizzando la AWS CLI, sono necessari i seguenti parametri:

- `cluster-name`: il nome del cluster.
- `k8s-version`: la versione di Kubernetes (ad esempio, 1.31).
- `subnet-ids`: ID di sottorete identificati nelle fasi precedenti.
- `cluster-role-arn`: ARN del ruolo IAM del cluster.
- `node-role-arn`: ARN del ruolo IAM del nodo.

## Configurazioni predefinite del cluster

Esaminare questi valori e funzionalità predefiniti prima di creare il cluster:

- `nodePools`: modalità automatica di EKS include pool di nodi generici e predefiniti di sistema. Maggiori informazioni sui Node Pools.

Nota: i pool di nodi in modalità automatica di EKS differiscono dai gruppi di nodi gestiti di Amazon EKS ma possono coesistere nello stesso cluster.

- `computeConfig.enabled`: automatizza le attività di elaborazione di routine, come la creazione e l'eliminazione di istanze EC2.
- `kubernetesNetworkConfig.elasticLoadBalancing.enabled`: automatizza le attività di bilanciamento del carico, includendo la creazione e l'eliminazione dei bilanciatori del carico elastico.
- `storageConfig.blockStorage.enabled`: automatizza le attività di archiviazione, come la creazione e l'eliminazione di volumi Amazon EBS.
- `accessConfig.authenticationMode`: richiede voci di accesso EKS. Maggiori informazioni su EKS authentication modes.

## Impostazione delle variabili di ambiente

Prima di eseguire il `create-cluster` comando, impostate le seguenti variabili di ambiente. Sostituisci i valori segnaposto con i tuoi:

```
export AWS_REGION=us-east-1
export CLUSTER_NAME=my-auto-cluster
export K8S_VERSION=1.35
export CLUSTER_ROLE_ARN=arn:aws:iam::111122223333:role/AmazonEKSAutoClusterRole
export NODE_ROLE_ARN=arn:aws:iam::111122223333:role/AmazonEKSAutoNodeRole
export SUBNETS_JSON='["subnet-ExampleID1","subnet-ExampleID2","subnet-ExampleID3"]'
```

> **Note**
>
> Imposta la SUBNETS_JSON variabile con gli ID di sottorete del passaggio precedente. È necessario racchiudere il valore tra virgolette singole per evitare che la shell interpreti le parentesi e le virgolette come caratteri speciali.

Verifica le variabili (opzionale)

Verifica che le variabili siano impostate correttamente:

```
echo "Region: ${AWS_REGION}"
echo "Cluster Name: ${CLUSTER_NAME}"
echo "K8s Version: ${K8S_VERSION}"
echo "Cluster Role ARN: ${CLUSTER_ROLE_ARN}"
echo "Node Role ARN: ${NODE_ROLE_ARN}"
echo "Subnets JSON: ${SUBNETS_JSON}"
```

Esegui il comando

Usa il comando seguente per creare il cluster:

```
aws eks create-cluster \
  --region ${AWS_REGION} \
  --cli-input-json \
  "{
      \"name\": \"${CLUSTER_NAME}\",
      \"version\": \"${K8S_VERSION}\",
      \"roleArn\": \"${CLUSTER_ROLE_ARN}\",
      \"resourcesVpcConfig\": {
        \"subnetIds\": ${SUBNETS_JSON},
        \"endpointPublicAccess\": true,
        \"endpointPrivateAccess\": true
      },
```

```
    \"computeConfig\": {
      \"enabled\": true,
      \"nodeRoleArn\":\"${NODE_ROLE_ARN}\",
      \"nodePools\": [\"general-purpose\", \"system\"]
    },
    \"kubernetesNetworkConfig\": {
      \"elasticLoadBalancing\": {
        \"enabled\": true
      }
    },
    \"storageConfig\": {
      \"blockStorage\": {
        \"enabled\": true
      }
    },
    \"accessConfig\": {
      \"authenticationMode\": \"API\"
    }
  }"
```

Verifica lo stato del cluster

Fase 1: verificare la creazione del cluster

Esegui il comando seguente per verificare lo stato del cluster. La creazione del cluster richiede in genere 15 minuti.

```
aws eks describe-cluster --name "${CLUSTER_NAME}" --output json
```

Fase 2: aggiornare kubeconfig

Una volta che il cluster è pronto, aggiorna il file kubeconfig locale per consentire a `kubectl` di comunicare con il cluster. Questa configurazione utilizza la AWS CLI per l'autenticazione.

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Fase 3: verificare i pool di nodi

Elenca i pool di nodi nel cluster con il comando seguente:

```
kubectl get nodepools
```

### Fasi successive

- Informazioni su deploy a sample workload nel nuovo cluster della modalità automatica di EKS.

## Crea un cluster EKS Auto Mode con Console di gestione AWS

La creazione di un cluster EKS Auto Mode in Console di gestione AWS formato richiede meno configurazioni rispetto ad altre opzioni. EKS si integra con AWS IAM e VPC Networking per aiutarti a creare le risorse associate a un cluster EKS.

Hai due opzioni per creare un cluster nella console:

- Configurazione rapida (con la modalità automatica di EKS)
- Configurazione personalizzata

Questo argomento illustra come creare un cluster della modalità automatica di EKS utilizzando l'opzione di configurazione rapida.

### Crea un cluster EKS Auto Mode utilizzando l'opzione di configurazione rapida

È necessario accedere Console di gestione AWS con autorizzazioni sufficienti per gestire le AWS risorse, tra cui: istanze EC2, reti EC2, cluster EKS e ruoli IAM.

1. Passa alla console EKS
2. Scegli Crea cluster
3. Conferma che l'opzione di configurazione rapida è selezionata
4. Determina i seguenti valori o usare i valori predefiniti per un cluster di test.
   - Nome del cluster
   - Versione di Kubernetes
5. Seleziona il ruolo IAM del cluster. Se è la prima volta che hai creato un cluster della modalità automatica di EKS, utilizza l'opzione Crea ruolo consigliato.
   - Facoltativamente, puoi riutilizzare un singolo ruolo IAM del cluster nel tuo AWS account per tutti i cluster EKS Auto Mode.
   - Il ruolo IAM del cluster include le autorizzazioni necessarie affinché modalità automatica di EKS gestisca le risorse, tra cui istanze EC2, volumi EBS e bilanciatori del carico EC2.

- L'opzione Crea ruolo consigliato precompila tutti i campi con i valori consigliati. Seleziona Avanti e poi Crea. Il ruolo utilizzerà il nome `AmazonEKSAutoClusterRole` suggerito.
- Se recentemente hai creato un nuovo ruolo, usa l'icona Aggiorna per ricaricare il menu a discesa di selezione del ruolo.

6. Seleziona il ruolo IAM del nodo. Se è la prima volta che hai creato un cluster della modalità automatica di EKS, utilizza l'opzione Crea ruolo consigliato.
   - Facoltativamente, puoi riutilizzare un singolo ruolo IAM del nodo nel tuo AWS account per tutti i cluster EKS Auto Mode.
   - Il ruolo IAM del nodo include le autorizzazioni necessarie per la connessione dei nodi Auto Mode al cluster. Il ruolo IAM del nodo deve includere le autorizzazioni per recuperare le immagini ECR per i container.
   - L'opzione Crea ruolo consigliato precompila tutti i campi con i valori consigliati. Seleziona Avanti e poi Crea. Il ruolo utilizzerà il nome `AmazonEKSAutoNodeRole` suggerito.
   - Se recentemente hai creato un nuovo ruolo, usa l'icona Aggiorna per ricaricare il menu a discesa di selezione del ruolo.
7. Seleziona il VPC per il cluster della modalità automatica di EKS. Scegli Crea VPC per creare un nuovo VPC per EKS oppure scegli un VPC creato in precedenza per EKS.
   - Se utilizzi la console VPC per creare un nuovo VPC, ti AWS suggerisce di creare almeno un gateway NAT per zona di disponibilità. In caso contrario, puoi usare tutte le altre impostazioni predefinite.
   - Per ulteriori informazioni e dettagli sui requisiti del cluster IPv6, consulta the section called "Crea un VPC".
8. (opzionale) modalità automatica di EKS popola automaticamente le sottoreti private per il VPC selezionato. Puoi rimuovere le sottoreti indesiderate.
   - EKS seleziona automaticamente le sottoreti private dal VPC seguendo le migliori pratiche. Facoltativamente, puoi selezionare sottoreti aggiuntive dal VPC, come quelle pubbliche.
9. (facoltativo) Seleziona Visualizza impostazioni predefinite di configurazione rapida per esaminare tutti i valori di configurazione per il nuovo cluster. La tabella indica che alcuni valori non sono modificabili dopo la creazione del cluster.
10. Selezionare Creazione di un cluster. Il completamento della creazione del cluster potrebbe richiedere quindici minuti.

### Fasi successive

- Informazioni su Deploy a Sample Workload to your EKS Auto Mode cluster

## Abilita la modalità automatica di EKS sui cluster EKS esistenti

Puoi abilitare la modalità automatica di EKS sui cluster EKS esistenti.

AWS supporta le seguenti migrazioni:

- Migrazione da Karpenter ai nodi della modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Eseguire la migrazione da Karpenter".
- Migrazione dai gruppi di nodi gestiti EKS ai nodi della modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Eseguire la migrazione da MNG".
- Migrazione da EKS Fargate alla modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Eseguire la migrazione da Fargate".

AWS non supporta le seguenti funzionalità:

- Migrazione di volumi dal controller CSI EBS (utilizzando il componente aggiuntivo Amazon EKS) al controller CSI EBS di la modalità automatica di EKS (gestito dalla modalità automatica di EKS). I PVC realizzati con uno non possono essere montati con l'altro, perché utilizzano due diversi provider di volumi Kubernetes.
  - Il `eks-auto-mode-ebs-migration-tool`(progetto AWS Labs) consente la migrazione tra EBS CSI standard StorageClass (`ebs.csi.aws.com`) e EKS Auto EBS CSI (). StorageClass `ebs.csi.eks.amazonaws.com` Tieni presente che la migrazione richiede l'eliminazione e la ricreazione PersistentVolumeClaim/PersistentVolume delle risorse esistenti, quindi la convalida in un ambiente non di produzione è essenziale prima dell'implementazione.
- Migrazione dei sistemi di bilanciamento del carico dal AWS Load Balancer Controller alla modalità automatica EKS

  Puoi installare il AWS Load Balancer Controller su un cluster Amazon EKS Auto Mode. Utilizza le opzioni `IngressClass` o `loadBalancerClass` per associare le risorse Service e Ingress al Load Balancer Controller o alla modalità automatica di EKS. Per una guida prescrittiva, consulta `Migrate NGINX Ingress Controllers when enabling Amazon EKS Auto Mode`
- Migrazione di cluster EKS con CNI alternativi o altre configurazioni di rete non supportate

# Materiale di riferimento per le migrazioni

Utilizza il seguente riferimento sulla migrazione per configurare le risorse Kubernetes in modo che siano di proprietà di controller autogestiti o la modalità automatica di EKS.

| Funzionalità | Risorsa | Campo | Autogestito | Modalità automatica di EKS |
|---|---|---|---|---|
| Storage a blocchi | `StorageClass` | `provisioner` | `ebs.csi.aws.com` | `ebs.csi.eks.amazonaws.com` |
| Sistema di bilanciamento del carico | `Service` | `loadBalancerClass` | `service.k8s.aws/nlb` | `eks.amazonaws.com/nlb` |
| Sistema di bilanciamento del carico | `IngressClass` | `controller` | `ingress.k8s.aws/alb` | `eks.amazonaws.com/alb` |
| Sistema di bilanciamento del carico | `IngressClassParams` | `apiversion` | `elbv2.k8s.aws/v1beta1` | `eks.amazonaws.com/v1` |
| Sistema di bilanciamento del carico | `TargetGroupBinding` | `apiversion` | `elbv2.k8s.aws/v1beta1` | `eks.amazonaws.com/v1` |
| Calcolo | `NodeClass` | `apiVersion` | `karpenter.sh/v1` | `eks.amazonaws.com/v1` |

# Migration di volumi EBS

Durante la migrazione dei carichi di lavoro alla modalità automatica di EKS, è necessario gestire la migrazione dei volumi EBS a causa di diversi provisioner di driver CSI:

- Provisioner della modalità automatica di EKS: `ebs.csi.eks.amazonaws.com`
- Provisioner CSI EBS open source: `ebs.csi.aws.com`

Segui questi passaggi per migrare i volumi persistenti:

1. Modifica la politica di conservazione del volume: modifica PersistentVolume quella esistente (PV) `persistentVolumeReclaimPolicy` per `Retain` garantire che il volume EBS sottostante non venga eliminato.
2. Rimuovi la versione della piattaforma da Kubernetes: elimina la vecchia risorsa della versione mantenendo intatto il volume EBS effettivo.
3. Crea una nuova versione della piattaforma con provisioning statico: crea una nuova versione che faccia riferimento allo stesso volume EBS ma funzioni con il driver CSI di destinazione.
4. Associa a un nuovo PVC: crea un nuovo PVC che faccia riferimento in modo specifico alla versione della tua piattaforma utilizzando il campo `volumeName`.

### Considerazioni

- Assicurati che le tue applicazioni siano interrotte prima di iniziare questa migrazione.
- Esegui il backup dei dati prima di iniziare il processo di migrazione.
- Questo processo deve essere eseguito per ogni volume persistente.
- Il carico di lavoro deve essere aggiornato per utilizzare il nuovo PVC.

## Migrazione dei bilanciatori del carico

Non è possibile trasferire direttamente i sistemi di bilanciamento del carico esistenti dal controller di bilanciamento del AWS carico autogestito alla modalità automatica EKS. Devi invece implementare una strategia di implementazione blu/verde. Questo comporta il mantenimento della configurazione del bilanciatore del carico esistente e la creazione di nuovi sistemi di bilanciamento del carico sotto il controller gestito.

Per ridurre al minimo le interruzioni del servizio, consigliamo un approccio basato sullo spostamento del traffico. DNS-based Innanzitutto, crea nuovi sistemi di bilanciatore del carico utilizzando la modalità automatica di EKS e mantenendo operativa la configurazione esistente. Quindi, utilizza il routing DNS (come Route 53) per spostare gradualmente il traffico dai vecchi sistemi di bilanciatore

del carico a quelli nuovi. Una volta completata la migrazione del traffico e verificata la nuova configurazione, puoi disattivare i vecchi sistemi di bilanciatore del carico e il controller autogestito.

## Abilita modalità automatica di EKS su un cluster esistente

In questo argomento viene descritto come abilitare la modalità automatica di Amazon EKS sui cluster Amazon EKS esistenti. L'attivazione di Auto Mode su un cluster esistente richiede l'aggiornamento delle autorizzazioni IAM e la configurazione delle impostazioni principali della modalità automatica di EKS. Dopo l'abilitazione, puoi iniziare a eseguire la migrazione dei carichi di lavoro di elaborazione esistenti per sfruttare le operazioni semplificate e la gestione automatizzata dell'infrastruttura di Auto Mode.

> **Important**
>
> Verifica di avere Add-ons installato la versione minima richiesta di alcuni Amazon EKS prima di attivare la modalità automatica EKS. Per ulteriori informazioni, consulta the section called "Versioni richieste del componente aggiuntivo".

Prima di iniziare, assicurati di disporre dell'accesso da amministratore al cluster Amazon EKS e delle autorizzazioni per modificare i ruoli IAM. I passaggi descritti in questo argomento ti guidano nell'attivazione della modalità automatica utilizzando Console di gestione AWS o la AWS CLI.

### Console di gestione AWS

È necessario accedere alla AWS console con l'autorizzazione per gestire le risorse IAM, EKS ed EC2.

> **Note**
>
> Il ruolo IAM del cluster di un cluster EKS non può essere modificato dopo la creazione del cluster. modalità automatica di EKS richiede autorizzazioni aggiuntive per questo ruolo. Devi allegare policy aggiuntive al ruolo attuale.

Aggiorna il ruolo IAM del cluster

1. Apri la pagina di panoramica del cluster in Console di gestione AWS.

2. In ARN del ruolo IAM del cluster, seleziona Visualizza in IAM.
3. Dall'elenco a discesa Aggiungi autorizzazioni, scegli Allega policy.
4. Utilizza la casella Ricerca per trovare e selezionare le seguenti policy:
   - `AmazonEKSComputePolicy`
   - `AmazonEKSBlockStoragePolicyV2`
   - `AmazonEKSLoadBalancingPolicy`
   - `AmazonEKSNetworkingPolicy`
   - `AmazonEKSClusterPolicy`
5. Seleziona Aggiungi autorizzazioni.
6. Dalla scheda Relazioni di attendibilità, seleziona Modifica policy di attendibilità
7. Inserisci la seguente policy di attendibilità del ruolo IAM del cluster e seleziona Aggiorna policy

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Abilita modalità automatica di EKS

1. Apri la pagina di panoramica del cluster in Console di gestione AWS.
2. In modalità automatica di EKS, seleziona Gestisci
3. Attiva modalità automatica di EKS.
4. Dal menu a discesa Pool di nodi EKS, seleziona i pool di nodi predefiniti da creare.
   - Maggiori informazioni sui pool di nodi in modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

5. Se in precedenza hai creato un ruolo IAM EKS Auto Mode Node in questo AWS account, selezionalo nel menu a discesa  Node IAM Role. Se non hai mai creato questo ruolo in precedenza, seleziona Crea ruolo consigliato e seguire i passaggi.

## AWS CLI

### Prerequisiti

- Il ruolo IAM del cluster del cluster EKS esistente deve includere autorizzazioni sufficienti per EKS Auto Mode, come le seguenti politiche:
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicyV2`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`
- Il ruolo IAM del cluster deve avere una policy di attendibilità aggiornata che includa l'operazione `sts:TagSession`. Per ulteriori informazioni sulla creazione di un ruolo IAM del cluster, consulta the section called " AWS CLI".
- `aws`CLI installata, loggata e una versione sufficiente. Devi disporre dell'autorizzazione per gestire le risorse IAM, EKS ed EC2. Per ulteriori informazioni, consulta Configurazione.

### Procedura

Utilizza i seguenti comandi per abilitare modalità automatica di EKS su un cluster esistente.

> **Note**
>
> Le funzionalità di calcolo, archiviazione a blocchi e bilanciamento del carico devono essere tutte abilitate o disabilitate nella stessa richiesta.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=true \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": true}}' \
 --storage-config '{"blockStorage":{"enabled": true}}'
```

## Versioni richieste del componente aggiuntivo

Se intendi abilitare modalità automatica di EKS su un cluster esistente, potrebbe dover aggiornare alcuni componenti aggiuntivi. Nota:

- Questo vale solo per i cluster esistenti che passano alla modalità automatica di EKS .
- I nuovi cluster creati con la modalità automatica di EKS abilitata non richiedono questi aggiornamenti.

Se hai installato uno dei seguenti componenti aggiuntivi, assicurati che abbia almeno la versione minima specificata:

| Add-on nome | Versione minima richiesta |
|---|---|
| Plug-in della CNI di Amazon VPC per Kubernetes | v1.19.0-eksbuild.1 |
| Kube-proxy | • v1.26.15-eksbuild.19<br>• v1.27.16-eksbuild.14<br>• v1.28.15-eksbuild.4<br>• v1.29.10-eksbuild.3<br>• v1.30.6-eksbuild.3<br>• v1.31.2-eksbuild.3 |
| Driver CSI per Amazon EBS | v1.37.0-eksbuild.1 |
| Controller di snapshot CSI | v8.1.0-eksbuild.2 |
| EKS Pod Identity Agent | v1.3.4-eksbuild.1 |

Per ulteriori informazioni, consulta the section called "Aggiornamento di un componente aggiuntivo".

## Fasi successive

- Per migrare i carichi di lavoro di Managed Node Group, consulta. the section called "Eseguire la migrazione da MNG"

- Per migrare da Self-Managed Karpenter, consulta. the section called "Eseguire la migrazione da Karpenter"

## Eseguire la migrazione da Karpenter alla modalità automatica di EKS utilizzando kubectl

Questo argomento illustra il processo di migrazione dei carichi di lavoro da Karpenter alla modalità automatica di Amazon EKS utilizzando kubectl. La migrazione può essere eseguita gradualmente, consentendo di spostare i carichi di lavoro al proprio ritmo, mantenendo al contempo la stabilità del cluster e la disponibilità delle applicazioni durante la transizione.

L'approccio dettagliato descritto di seguito ti consente di eseguire Karpenter ed modalità automatica di EKS in parallelo durante il periodo di migrazione. Questa strategia a doppio funzionamento aiuta a garantire una transizione fluida consentendoti di convalidare il comportamento del carico di lavoro in modalità automatica di EKS prima di disattivare completamente Karpenter. Puoi eseguire la migrazione delle applicazioni singolarmente o in gruppo, garantendo la flessibilità necessaria per soddisfare i requisiti operativi specifici e la tolleranza al rischio.

### Prerequisiti

Prima di iniziare la migrazione, assicurati di avere:

- Karpenter v1.1 o versioni successive deve essere installato sul cluster. Per ulteriori informazioni, consulta Upgrading to 1.1.0+ nei documenti di Karpenter.
- `kubectl` installato e connesso al cluster. Per ulteriori informazioni, consulta Configurazione.

Questo argomento presuppone che tu conosca Karpenter e. NodePools Per ulteriori informazioni, consulta Karpenter Documentation.

> **Note**
>
> Alcuni CRD (ad esempio, `nodepools.karpenter.sh` `enodeclaims.karpenter.sh`) sono condivisi tra EKS Auto Mode e Karpenter. Non aggiornate o modificate questi CRD durante la migrazione.

### Fase 1: abilitare modalità automatica di EKS sul cluster

Abilita la modalità automatica EKS sul cluster esistente utilizzando la AWS CLI o la console di gestione. Per ulteriori informazioni, consulta the section called "Abilitazione sul cluster".

> **Note**
>
> Durante l'abilitazione della modalità automatica di EKS, non abilitare il nodepool `general purpose` in questa fase durante la transizione. Questo pool di nodi non è selettivo.
> Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

### Fase 2: Creare una modalità EKS Auto contaminata NodePool

Creane una nuova NodePool per EKS Auto Mode con una macchia. Ciò garantisce che i pod esistenti non vengano programmati automaticamente sui nuovi nodi della modalità automatica di EKS. Questo pool di nodi utilizza `NodeClass` integrato `default` in modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

Esempio di pool di nodi con taint:

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      taints:
        - key: "eks-auto-mode"
          effect: "NoSchedule"
```

Aggiorna i requisiti per il pool di nodi in modo tale che corrispondano alla configurazione di Karpenter da cui stai eseguendo la migrazione. Devi soddisfare almeno un requisito.

### Fase 3: aggiornare i carichi di lavoro per la migrazione

Identifica e aggiorna i carichi di lavoro che desideri migrare alla modalità automatica di EKS . Aggiungi sia le tolleranze che i selettori di nodi a questi carichi di lavoro:

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

Questa modifica consente di pianificare il carico di lavoro sui nuovi nodella modalità automatica di EKS.

La modalità automatica di EKS utilizza etichette diverse rispetto a Karpenter. Le etichette relative alle istanze gestite da EC2 iniziano con `eks.amazonaws.com`. Per ulteriori informazioni, consulta the section called “Creazione di pool di nodi”.

### Fase 4: migrare gradualmente i carichi di lavoro

Ripeti la fase 3 per ogni carico di lavoro di cui desideri eseguire la migrazione. Ciò consente di spostare i carichi di lavoro individualmente o in gruppo, in base ai requisiti e alla tolleranza al rischio.

### Fase 5: Rimuovere il Karpenter originale NodePool

Una volta migrati tutti i carichi di lavoro, puoi rimuovere il Karpenter originale: NodePool

```
kubectl delete nodepool <original-nodepool-name>
```

### Passaggio 6: rimuovere la macchia dalla modalità automatica EKS (opzionale) NodePool

Se desideri che EKS Auto Mode diventi l'impostazione predefinita per i nuovi carichi di lavoro, puoi rimuovere la macchia dalla modalità EKS Auto: NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      # Remove the taints section
```

### Fase 7: rimuovere i selettori di nodi dai carichi di lavoro (facoltativo)

Se hai rimosso la macchia dalla modalità EKS Auto NodePool, puoi opzionalmente rimuovere i selettori di nodo dai tuoi carichi di lavoro, poiché EKS Auto Mode è ora l'impostazione predefinita:

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      # Remove the nodeSelector section
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
```

### Fase 8: disinstallare Karpenter dal cluster

Le fasi per rimuovere Karpenter dipendono da come l'hai installato. Per ulteriori informazioni, consulta Karpenter install instructions.

# Eseguire la migrazione dai gruppi di nodi gestiti EKS a EKS Auto Mode

Quando si esegue la transizione del cluster Amazon EKS per utilizzare EKS Auto Mode, puoi eseguire la migrazione senza problemi i carichi di lavoro esistenti dai gruppi di nodi gestiti (MNG) utilizzando lo strumento CLI eksctl. Questo processo garantisce la disponibilità continua delle applicazioni mentre EKS Auto Mode ottimizza le risorse di calcolo. La migrazione può essere eseguita con interruzioni minime delle applicazioni in esecuzione.

Questo argomento illustra le fasi per drenare in sicurezza i pod dai gruppi di nodi gestiti esistenti e consentire a EKS Auto Mode di riprogrammarli sulle nuove istanze di cui è appena stato effettuato il provisioning. Seguendo questa procedura, puoi sfruttare il consolidamento intelligente del carico di lavoro di EKS Auto Mode mantenendo al contempo la disponibilità dell'applicazione durante la migrazione.

## Prerequisiti

- Cluster con EKS Auto Mode abilitata
- `eksctl` CLI installato e connesso al cluster. Per ulteriori informazioni, consulta Configurazione.
- Karpenter non è installato nel cluster.

## Procedura

Utilizza il seguente comando `eksctl` CLI per avviare i pod di drenaggio dalle istanze del gruppo di nodi gestiti esistente. EKS Auto Mode creerà nuovi nodi per supportare i pod spostati.

```
eksctl delete nodegroup --cluster=<clusterName> --name=<nodegroupName>
```

Dovrai eseguire questo comando per ogni gruppo di nodi gestiti nel cluster.

Per ulteriori informazioni su questo comando, consulta Deleting and draining nodegroups nei documenti di eksctl.

# Eseguire la migrazione da EKS Fargate alla modalità automatica di EKS

Questo argomento descrive il processo di migrazione dei carichi di lavoro da EKS Fargate alla modalità automatica di Amazon EKS utilizzando `kubectl`. La migrazione può essere eseguita gradualmente, consentendo di spostare i carichi di lavoro al proprio ritmo, mantenendo al contempo la stabilità del cluster e la disponibilità delle applicazioni durante la transizione.

L'approccio dettagliato descritto di seguito ti consente di eseguire EKS Fargate ed modalità automatica di EKS in parallelo durante il periodo di migrazione. Questa strategia a doppio funzionamento aiuta a garantire una transizione fluida consentendo di convalidare il comportamento del carico di lavoro in modalità automatica di EKS prima di disattivare completamente EKS Fargate. Puoi eseguire la migrazione delle applicazioni singolarmente o in gruppo, garantendo la flessibilità necessaria per soddisfare i requisiti operativi specifici e la tolleranza al rischio.

## Confronto tra Amazon EKS Auto Mode ed EKS con AWS Fargate

Amazon EKS con AWS Fargate rimane un'opzione per i clienti che desiderano utilizzare EKS, ma Amazon EKS Auto Mode è l'approccio consigliato per il futuro. La modalità automatica di EKS è completamente conforme a Kubernetes e supporta tutte le primitive Kubernetes a monte e gli strumenti di piattaforma come Istio, che Fargate non è in grado di supportare. Inoltre, modalità automatica di EKS supporta completamente tutte le opzioni di acquisto del runtime EC2, incluse GPU e istanze Spot, consentendo ai clienti di sfruttare gli sconti EC2 negoziati e altri meccanismi di risparmio. Queste funzionalità non sono disponibili quando si utilizza EKS con Fargate.

Inoltre, modalità automatica di EKS consente ai clienti di ottenere lo stesso modello di isolamento di Fargate, utilizzando le funzionalità di pianificazione di Kubernetes standard per garantire che ogni istanza EC2 esegua un singolo container di applicazioni. Adottando Amazon EKS Auto Mode, i clienti possono sfruttare tutti i vantaggi dell'esecuzione di Kubernetes su AWS : una Kubernetes-conformant piattaforma completa che offre la flessibilità necessaria per sfruttare l'intera gamma di EC2 e le opzioni di acquisto, pur mantenendo la facilità d'uso e l'astrazione dalla gestione dell'infrastruttura offerte da Fargate.

Raggiungere l'isolamento in modalità EKS Auto Fargate-like

Per replicare il modello di isolamento dei pod di Fargate in cui ogni pod viene eseguito su una propria istanza dedicata, puoi utilizzare i vincoli di diffusione della topologia di Kubernetes. Questo è l'approccio consigliato per controllare la distribuzione dei pod tra i nodi:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: isolated-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: isolated-app
```

```
  template:
    metadata:
      labels:
        app: isolated-app
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: isolated-app
        minDomains: 1
      containers:
      - name: app
        image: nginx
        ports:
        - containerPort: 80
```

In questa configurazione:

- `maxSkew: 1`assicura che la differenza nel numero di pod tra due nodi qualsiasi sia al massimo 1, distribuendo efficacemente un pod per nodo
- `topologyKey: kubernetes.io/hostname`definisce il nodo come dominio topologico
- `whenUnsatisfiable: DoNotSchedule`impedisce la pianificazione se il vincolo non può essere soddisfatto
- `minDomains: 1`garantisce l'esistenza di almeno un dominio (nodo) prima della pianificazione

EKS Auto Mode fornirà automaticamente nuove istanze EC2 secondo necessità per soddisfare questo vincolo, fornendo lo stesso modello di isolamento di Fargate e dandovi accesso all'intera gamma di tipi di istanze EC2 e opzioni di acquisto.

In alternativa, puoi utilizzare le regole anti-affinità dei pod per un isolamento più rigoroso:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: isolated-app
spec:
```

```
  replicas: 3
  selector:
    matchLabels:
      app: isolated-app
  template:
    metadata:
      labels:
        app: isolated-app
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - isolated-app
            topologyKey: kubernetes.io/hostname
      containers:
      - name: app
        image: nginx
        ports:
        - containerPort: 80
```

La `podAntiAffinity` regola con `requiredDuringSchedulingIgnoredDuringExecution` assicura che non `app: isolated-app` possano essere programmati due pod con l'etichetta sullo stesso nodo. Questo approccio fornisce garanzie di isolamento rigide simili a Fargate.

## Prerequisiti

Prima di iniziare la migrazione, assicurati di avere:

- configurato un cluster con Fargate. Per ulteriori informazioni, consulta the section called "Nozioni di base";
- installato e connesso `kubectl` al cluster. Per ulteriori informazioni, consulta Configurazione.

## Fase 1: Verificare il cluster Fargate

1. Verifica se il cluster EKS con Fargate è in esecuzione:

```
kubectl get node
```

```
NAME STATUS ROLES AGE VERSION
fargate-ip-192-168-92-52.ec2.internal Ready <none> 25m v1.30.8-eks-2d5f260
fargate-ip-192-168-98-196.ec2.internal Ready <none> 24m v1.30.8-eks-2d5f260
```

2. Verifica i pod in esecuzione:

```
kubectl get pod -A
```

```
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system coredns-6659cb98f6-gxpjz 1/1 Running 0 26m
kube-system coredns-6659cb98f6-gzzsx 1/1 Running 0 26m
```

3. Create un'implementazione in un file chiamato `deployment_fargate.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: fargate
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

4. Applica l'implementazione:

```
kubectl apply -f deployment_fargate.yaml
```

```
deployment.apps/nginx-deployment created
```

5. Verifica i pod e le implementazioni:

```
kubectl get pod,deploy
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-5c7479459b-6trtm   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-g8ssb   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-mq4mf   1/1     Running   0          61s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           61s
```

6. Verifica il nodo:

```
kubectl get node -owide
```

```
NAME                                      STATUS  ROLES  AGE VERSION
   INTERNAL-IP     EXTERNAL-IP OS-IMAGE       KERNEL-VERSION
 CONTAINER-RUNTIME
fargate-ip-192-168-111-43.ec2.internal  Ready   <none> 31s v1.30.8-eks-2d5f260
 192.168.111.43  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-117-130.ec2.internal Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.117.130 <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-74-140.ec2.internal  Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.74.140  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
```

## Fase 2: Abilitare modalità automatica di EKS sul cluster

1. Abilita la modalità automatica EKS sul tuo cluster esistente utilizzando la AWS CLI o la console di gestione. Per ulteriori informazioni, consulta the section called "Abilitazione sul cluster".
2. Verifica il nodepool:

```
kubectl get nodepool
```

```
NAME              NODECLASS   NODES   READY   AGE
general-purpose   default     1       True    6m58s
system            default     0       True    3d14h
```

## Fase 3: Aggiornare i carichi di lavoro per la migrazione

Identifica e aggiorna i carichi di lavoro che desideri migrare alla modalità automatica di EKS .

Per eseguire la migrazione di un carico di lavoro da Fargate alla modalità automatica di EKS , applica l'annotazione `eks.amazonaws.com/compute-type: ec2`. Ciò garantisce che il carico di lavoro non venga pianificato da Fargate, nonostante il profilo Fargate, e venga recuperato dalla modalità EKS Auto. NodePool Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

1. Modifica le implementazioni (ad esempio, il file `deployment_fargate.yaml`) per cambiare il tipo di elaborazione in `ec2`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
```

```
        - containerPort: 80
```

2. Applica l'implementazione. Questa modifica consente di pianificare il carico di lavoro sui nuovi nodella modalità automatica di EKS:

```
kubectl apply -f deployment_fargate.yaml
```

3. Verifica che l'implementazione sia in esecuzione nel cluster della modalità automatica di EKS:

```
kubectl get pod -o wide
```

```
NAME                               READY   STATUS    RESTARTS   AGE     IP
    NODE                  NOMINATED NODE   READINESS GATES
nginx-deployment-97967b68d-ffxxh   1/1     Running   0          3m31s
 192.168.43.240   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-mbcgj   1/1     Running   0          2m37s
 192.168.43.241   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-qpd8x   1/1     Running   0          2m35s
 192.168.43.242   i-0845aafcb51630ffb   <none>           <none>
```

4. Verifica che non sia in esecuzione alcun nodo Fargate e che l'implementazione sia in esecuzione nei nodi gestiti dalla modalità automatica di EKS:

```
kubectl get node -owide
```

```
NAME                 STATUS ROLES  AGE   VERSION              INTERNAL-IP     EXTERNAL-
IP OS-IMAGE                                        KERNEL-VERSION CONTAINER-RUNTIME
i-0845aafcb51630ffb Ready  <none> 3m30s v1.30.8-eks-3c20087 192.168.41.125
 3.81.118.95 Bottlerocket (EKS Auto) 2025.3.14 (aws-k8s-1.30) 6.1.129
 containerd://1.7.25+bottlerocket
```

## Fase 4: Migrare gradualmente i carichi di lavoro

Ripeti la fase 3 per ogni carico di lavoro di cui desideri eseguire la migrazione. Ciò consente di spostare i carichi di lavoro individualmente o in gruppo, in base ai requisiti e alla tolleranza al rischio.

## Fase 5: Rimuovere il profilo Fargate originale

Una volta eseguita la migrazione di tutti i carichi di lavoro, puoi rimuovere il profilo `fargate` originale. Sostituisci *`<fargate profile name>`* con il nome del tuo profilo Fargate:

```
aws eks delete-fargate-profile --cluster-name eks-fargate-demo-cluster --fargate-
profile-name <fargate profile name>
```

### Fase 6: Ridurre verticalmente CoreDNS

Poiché EKS Auto Mode gestisce CoreDNS, ridimensiona la `coredns` distribuzione a 0:

```
kubectl scale deployment coredns -n kube-system --replicas=0
```

# Esegui carichi di lavoro di esempio nei cluster della modalità automatica di EKS

Questo capitolo fornisce esempi di come implementare diversi tipi di carichi di lavoro nei cluster Amazon EKS in esecuzione in Auto Mode. Gli esempi illustrano i principali schemi di carico di lavoro, tra cui applicazioni di esempio, applicazioni web con bilanciamento del carico, carichi di lavoro stateful che utilizzano l'archiviazione persistente, e carichi di lavoro con requisiti di posizionamento dei nodi specifici. Ogni esempio include manifesti completi e istruzioni di step-by-step distribuzione che è possibile utilizzare come modelli per le proprie applicazioni.

Prima di procedere con gli esempi, assicurati di avere un cluster EKS in esecuzione in modalità automatica e di aver installato la AWS CLI e kubectl. Per ulteriori informazioni, consulta Configurazione. Gli esempi presuppongono una conoscenza base dei concetti di Kubernetes e dei comandi kubectl.

Puoi utilizzare questi esempi basati su casi d'uso per eseguire carichi di lavoro in cluster della modalità automatica di EKS.

the section called "Implementazione di un carico di lavoro di decompressione"

Mostra come implementare un carico di lavoro di esempio in un cluster della modalità automatica di EKS utilizzando i comandi `kubectl`.

the section called "Implementa il bilanciatore del carico"

Mostra come implementare una versione containerizzata del gioco 2048 su Amazon EKS.

the section called "Implementazione di un carico di lavoro stateful"

Mostra come implementare un'applicazione stateful di esempio in un cluster della modalità automatica di EKS.

the section called "Implementare carico di lavoro accelerato"

Mostra come distribuire carichi di lavoro con accelerazione hardware nei nodi gestiti da EKS Auto Mode.

the section called "Controllare l'implementazione"

Mostra come utilizzare un'annotazione per controllare se un carico di lavoro viene implementato nei nodi gestiti dalla modalità automatica di EKS.

# Implementazione di un esempio di carico di lavoro di decompressione in un cluster con modalità automatica di Amazon EKS

In questo tutorial imparerai come implementare un carico di lavoro di esempio su un cluster con modalità automatica EKS e osservare come esegue automaticamente il provisioning delle risorse di calcolo necessarie. Utilizzerai `kubectl` i comandi per osservare il comportamento del cluster e vedere in prima persona come la modalità automatica semplifica le operazioni di Kubernetes su. AWS Al termine di questo tutorial, capirai come la modalità automatica EKS risponde alle distribuzioni dei carichi di lavoro gestendo automaticamente le risorse di calcolo sottostanti, senza richiedere la configurazione manuale dei gruppi di nodi.

## Prerequisiti

- Un cluster della modalità automatica di Amazon EKS. Annota il nome e la regione del cluster. AWS
- Un principale IAM, ad esempio un utente o un ruolo, con autorizzazioni sufficienti per gestire le risorse di rete, di elaborazione e EKS.
  - Per ulteriori informazioni, consulta Creating roles and attaching policies in the IAM User Guide nella Guida per l'utente IAM.
- `aws` CLI installata e configurata con un'identità IAM.
- `kubectl` CLI installata e connessa al cluster.
  - Per ulteriori informazioni, consulta Configurazione.

## Passaggio 1: rivedere le risorse di elaborazione esistenti (facoltativo)

Innanzitutto, usa `kubectl` per elencare i pool di nodi sul cluster.

```
kubectl get nodepools
```

Output di esempio:

```
general-purpose
```

In questo tutorial, implementeremo un carico di lavoro configurato per usare il pool di nodi `general-purpose`. Questo pool di nodi è integrato nella modalità automatica EKS e include impostazioni predefinite ragionevoli per carichi di lavoro generici, come microservizi e applicazioni web. Puoi creare un pool di nodi personalizzato. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

Poi, utilizza `kubectl` per elencare i nodi collegati al tuo cluster.

```
kubectl get nodes
```

Se hai appena creato un cluster con modalità automatica EKS, non avrai nodi.

In questo tutorial implementerai un carico di lavoro di esempio. Se non ci sono nodi o il carico di lavoro non può essere gestito dai nodi esistenti, la modalità automatica EKS provvederà a fornire un nuovo nodo.

## Passaggio 2: implementare un'applicazione di esempio sul cluster

Esamina la seguente implementazione di Kubernetes e salvala come `inflate.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate
  template:
    metadata:
      labels:
        app: inflate
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

```
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
      containers:
        - name: inflate
          image: public.ecr.aws/eks-distro/kubernetes/pause:3.7
          resources:
            requests:
              cpu: 1
          securityContext:
            allowPrivilegeEscalation: false
```

Il selettore `eks.amazonaws.com/compute-type: auto` richiede che il carico di lavoro sia implementato su un nodo della modalità automatica di Amazon EKS.

Applica l'implementazione al cluster.

```
kubectl apply -f inflate.yaml
```

### Passaggio 3: vedere gli eventi di Kubernetes

Utilizza il seguente comando per vedere gli eventi di Kubernetes, inclusa la creazione di un nuovo nodo. Usa `ctrl+c` per interrompere la visione degli eventi.

```
kubectl get events -w --sort-by '.lastTimestamp'
```

Utilizza `kubectl` per elencare nuovamente i nodi collegati al tuo cluster. Prendi nota del nodo appena creato.

```
kubectl get nodes
```

### Fase 4: Visualizza i nodi e le istanze in AWS console

È possibile visualizzare i nodi in modalità automatica EKS nella console EKS e le istanze EC2 associate nella console EC2.

Le istanze EC2 implementate dalla modalità automatica EKS sono limitate. Non puoi eseguire comandi arbitrari sui nodi della modalità automatica EKS.

### Passaggio 5: eliminare l'implementazione

Per eliminare un'implementazione, usa `kubectl`

```
kubectl delete -f inflate.yaml
```

Se non ci sono altri carichi di lavoro implementati nel cluster, il nodo creato dalla modalità automatica EKS sarà vuoto.

Nella configurazione predefinita, EKS Auto Mode rileva i nodi che sono rimasti vuoti per trenta secondi e li termina.

Usa `kubectl` o la console EC2 per confermare che l'istanza associata è stata eliminata.

## Implementa un carico di lavoro del bilanciatore del carico esemplificativo in modalità automatica di EKS

Questa guida mostra come implementare una versione containerizzata del gioco 2048 su Amazon EKS, completa di bilanciamento del carico e accessibilità a Internet.

### Prerequisiti

- Un cluster della modalità automatica di EKS
- `kubectl` configurato per interagire con il cluster
- Autorizzazioni IAM appropriate per la creazione di risorse ALB

### Fase 1: Creazione del namespace

Innanzitutto, crea un namespace dedicato per l'applicazione di gioco 2048.

Creare un file denominato `01-namespace.yaml`:

```
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
```

Applica la configurazione del namespace:

```
kubectl apply -f 01-namespace.yaml
```

## Fase 2: Implementazione dell'applicazione

L'applicazione esegue più repliche del container del gioco 2048.

Creare un file denominato `02-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
          resources:
            requests:
              cpu: "0.5"
```

> **Note**
>
> Se ricevi un errore durante il caricamento dell'immagine`public.ecr.aws/l6m2t8p7/docker-2048:latest`, conferma che il tuo ruolo Node IAM dispone di autorizzazioni sufficienti per estrarre immagini da ECR. Per ulteriori informazioni, consulta the section called "Ruolo IAM del nodo". Inoltre, l'immagine `docker-2048` nell'esempio è un'immagine `x86_64` e non sarà eseguita su altre architetture.

Componenti chiave:

- Implementa 5 repliche dell'applicazione
- Utilizza un'immagine ECR pubblica
- Richiede 0,5 core CPU per pod
- Espone la porta 80 per il traffico HTTP

Applica l'implementazione:

```
kubectl apply -f 02-deployment.yaml
```

## Fase 3: Creazione del servizio

Il servizio espone l'implementazione alla rete di cluster.

Creare un file denominato `03-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app.kubernetes.io/name: app-2048
```

Componenti chiave:

- Crea un servizio ClusterIP
- Mappa la porta 80 alla porta 80 del container
- Utilizza il selettore di etichette per trovare i pod

Applica il servizio:

```
kubectl apply -f 03-service.yaml
```

## Fase 4: Configurazione del bilanciamento del carico

Configurerai un ingresso per esporre l'applicazione a Internet.

Innanzitutto, crea la `IngressClass`. Creare un file denominato `04-ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  labels:
    app.kubernetes.io/name: LoadBalancerController
  name: alb
spec:
  controller: eks.amazonaws.com/alb
```

> **Note**
>
> La modalità automatica di EKS richiede i tag di sottorete per identificare le sottoreti pubbliche e private.
> Se hai creato il cluster con `eksctl`, disponi già di questi tag.
> Informazioni su come the section called "Assegnazione di tag alle sottoreti".

Poi, crea la risorsa Ingress. Creare un file denominato `05-ingress.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
```

```
            pathType: Prefix
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Componenti chiave:

- Crea un ALB connesso a Internet
- Utilizza il tipo di destinazione IP per l'instradamento diretto al pod
- Instrada tutto il traffico (/) al servizio di gioco

Applica le configurazioni di ingresso:

```
kubectl apply -f 04-ingressclass.yaml
kubectl apply -f 05-ingress.yaml
```

## Fase 5: Verifica dell'implementazione

1. Controlla che tutti i pod siano in esecuzione:

   ```
   kubectl get pods -n game-2048
   ```

2. Verifica che il servizio sia stato creato:

   ```
   kubectl get svc -n game-2048
   ```

3. Ottieni l'endpoint di ALB:

   ```
   kubectl get ingress -n game-2048
   ```

Il campo ADDRESS nell'output di ingresso mostrerà l'endpoint di ALB. Attendi 2-3 minuti affinché l'ALB effettui il provisioning e registri tutti gli obiettivi.

## Fase 6: Accesso al gioco

Apri il browser web e accedi all'URL dell'endpoint di ALB della fase precedente. Dovresti vedere l'interfaccia di gioco 2048.

### Fase 7: Eliminazione

Per rimuovere tutte le risorse create in questo tutorial:

```
kubectl delete namespace game-2048
```

Questo eliminerà tutte le risorse nel namespace, incluse le risorse di implementazione, di servizio e di ingresso.

### Cosa succede dietro le quinte

1. L'implementazione crea 5 pod che eseguono il gioco 2048
2. Il servizio fornisce un accesso stabile alla rete a questi pod
3. La modalità automatica di EKS:
   - Crea un Application Load Balancer in AWS
   - Configura i gruppi di destinazione per i pod
   - Imposta le regole di instradamento per indirizzare il traffico al servizio

### Risoluzione dei problemi

Se il gioco non si carica:

- Assicurati che tutti i pod siano in esecuzione: `kubectl get pods -n game-2048`
- Verifica lo stato di ingresso: `kubectl describe ingress -n game-2048`
- Verifica i controlli di integrità ALB: controlla lo stato del gruppo target nella console AWS

## Implementazione di un carico di lavoro stateful esemplificativo in modalità automatica EKS

Questo tutorial ti guiderà nella distribuzione di un'applicazione stateful esemplificativa nel tuo cluster di modalità automatica EKS. L'applicazione scrive marche temporali su un volume persistente, dimostrando le funzionalità automatiche di provisioning e persistenza dei volumi EBS di modalità automatica EKS.

### Prerequisiti

- Un cluster in modalità automatica EKS

- AWS CLI configurata con le autorizzazioni appropriate
- Installazione e configurazione di `kubectl`.
  - Per ulteriori informazioni, consulta Configurazione.

## Passaggio 1: configurazione dell'ambiente

1. Imposta le variabili di ambiente:

```
export CLUSTER_NAME=my-auto-cluster
export AWS_REGION="us-west-2"
```

2. Aggiorna kubeconfig:

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

## Passaggio 2: creazione della classe di archiviazione

`StorageClass` definisce in che modo la modalità automatica EKS fornirà i volumi EBS.

Modalità automatica EKS non crea un `StorageClass` per te. Devi creare una `StorageClass` che faccia riferimento a `ebs.csi.eks.amazonaws.com` per utilizzare la funzionalità di archiviazione di modalità automatica EKS.

1. Creare un file denominato `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Applica: `StorageClass`

```
kubectl apply -f storage-class.yaml
```

componenti chiave

- `provisioner: ebs.csi.eks.amazonaws.com`: utilizza modalità automatica EKS
- `volumeBindingMode: WaitForFirstConsumer`: ritarda la creazione del volume fino a quando un pod non ne ha bisogno
- `type: gp3`: specifica il tipo di volume EBS
- `encrypted: "true"`: EBS utilizzerà la chiave `aws/ebs` predefinita per crittografare i volumi creati con questa classe. Questo passaggio è facoltativo, ma è consigliato.
- `storageclass.kubernetes.io/is-default-class: "true"`: Kubernetes userà questa classe di archiviazione per impostazione predefinita, a meno che non sia specificata una classe di volume diversa su una dichiarazione di volume persistente. Fai attenzione quando imposti questo valore se stai migrando da un altro controller di archiviazione (facoltativo).

## Passaggio 3: creazione della dichiarazione di volume persistente

Il PVC richiede l'archiviazione da `StorageClass`.

1. Creare un file denominato `pvc.yaml`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-ebs-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: auto-ebs-sc
  resources:
    requests:
      storage: 8Gi
```

2. Applica il PVC:

```
kubectl apply -f pvc.yaml
```

Componenti chiave:

- `accessModes: ReadWriteOnce`: il volume può essere montato da un nodo alla volta
- `storage: 8Gi`: richiede un volume da 8 GiB
- `storageClassName: auto-ebs-sc`: riferimenti su `StorageClass` abbiamo creato

## Passaggio 4: implementare un'applicazione

L'implementazione esegue un container che scrive marche temporali nel volume persistente.

1. Creare un file denominato `deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate-stateful
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate-stateful
  template:
    metadata:
      labels:
        app: inflate-stateful
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      containers:
        - name: bash
          image: public.ecr.aws/docker/library/bash:4.4
          command: ["/usr/local/bin/bash"]
          args: ["-c", "while true; do echo $(date -u) >> /data/out.txt; sleep 60;
 done"]
          resources:
            requests:
              cpu: "1"
          volumeMounts:
            - name: persistent-storage
              mountPath: /data
      volumes:
```

```
        - name: persistent-storage
          persistentVolumeClaim:
            claimName: auto-ebs-claim
```

2. Applica l'implementazione:

```
kubectl apply -f deployment.yaml
```

Componenti chiave:

- Semplice container bash che scrive una marca temporale su un file
- Monta il PVC su `/data`
- Richiede 1 core CPU
- Utilizza il selettore di nodi per i nodi gestiti da EKS

## Passaggio 5: verificare la configurazione

1. Verifica che il pod sia in esecuzione:

```
kubectl get pods -l app=inflate-stateful
```

2. Verifica che il PVC sia legato:

```
kubectl get pvc auto-ebs-claim
```

3. Controlla il volume EBS:

```
# Get the PV name
PV_NAME=$(kubectl get pvc auto-ebs-claim -o jsonpath='{.spec.volumeName}')
# Describe the EBS volume
aws ec2 describe-volumes \
  --filters Name=tag:CSIVolumeName,Values=${PV_NAME}
```

4. Verifica che i dati siano scritti:

```
kubectl exec "$(kubectl get pods -l app=inflate-stateful \
  -o=jsonpath='{.items[0].metadata.name}')" -- \
  cat /data/out.txt
```

## Passaggio 6: pulizia

Eseguire il comando riportato di seguito per rimuovere tutte le risorse create in questo tutorial:

```
# Delete all resources in one command
kubectl delete deployment/inflate-stateful pvc/auto-ebs-claim storageclass/auto-ebs-sc
```

## Cosa succede dietro le quinte

1. Il PVC richiede l'archiviazione da `StorageClass`
2. Quando è il pod è pianificato:
   a. La modalità automatica EKS fornisce un volume EBS
   b. Crea PersistentVolume
   c. Collega il volume al nodo
3. Il pod monta il volume e inizia a scrivere le marche temporali

## Controller di snapshot

La modalità automatica EKS è compatibile con lo Snapshotter di CSI Kubernetes, noto anche come controller di snapshot. Tuttavia, la modalità automatica EKS non include il controller di snapshot. L'utente è responsabile dell'installazione e della configurazione del controller di snapshot. Per ulteriori informazioni, consulta the section called "Controller snapshot CSI".

Leggi quanto segue `VolumeSnapshotClass` che fa riferimento alla capacità di archiviazione della modalità automatica EKS.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: auto-ebs-vsclass
driver: ebs.csi.eks.amazonaws.com
deletionPolicy: Delete
```

Scopri di più su Snapshotter di CSI Kubernetes.

# Implementare un carico di lavoro accelerato

Questo tutorial dimostra come Amazon EKS Auto Mode semplifica l'avvio di carichi di lavoro con accelerazione hardware. Amazon EKS Auto Mode semplifica le operazioni che vanno oltre il cluster stesso automatizzando i componenti chiave dell'infrastruttura fornendo funzionalità di calcolo, rete, bilanciamento del carico, storage e gestione delle identità e degli accessi pronte all'uso.

Amazon EKS Auto Mode include i driver e i plug-in di dispositivo necessari per determinati tipi di istanze, come i driver NVIDIA e Neuron. AWS Non è necessario installare o aggiornare questi componenti.

modalità automatica di EKS gestisce automaticamente i driver per questi acceleratori:

- AWS Trainium
- AWS Inferenza
- GPU NVIDIA su istanze accelerate Amazon EC2

> **Note**
>
> La modalità automatica di EKS include il plug-in di dispositivo NVIDIA per Kubernetes. Questo plug-in viene eseguito automaticamente e non è visibile come daemon impostato nel cluster.

Supporto di reti avanzate:

- Elastic Fabric Adapter (EFA): per informazioni sulla gestione dei dispositivi EFA nel cluster EKS, consulta. the section called “Dispositivi EFA”

La modalità automatica di Amazon EKS elimina la difficoltà della gestione dei driver dell'acceleratore e dei plug-in di dispositivo.

Puoi anche trarre vantaggio dai risparmi sui costi ridimensionando il cluster fino a zero. Puoi configurare modalità automatica di EKS per terminare le istanze quando non è in esecuzione alcun carico di lavoro. Ciò è utile per i carichi di lavoro di inferenza basati su batch.

Di seguito viene fornito un esempio di come avviare carichi di lavoro accelerati con la modalità automatica di Amazon EKS.

## Prerequisiti

- Un cluster Kubernetes con la modalità automatica di Amazon EKS configurata.
- Una classe del nodo `default` EKS creata quando i pool di nodi gestiti da `general-purpose` o `system` sono abilitati.

## Fase 1: Implementare un carico di lavoro GPU

In questo esempio, creerai un file NodePool per carichi di lavoro basati su NVIDIA che richiede 45 GB di memoria GPU. Con la modalità automatica di EKS, utilizzi i vincoli di pianificazione di Kubernetes per definire i requisiti delle istanze.

Per implementare Amazon EKS Auto Mode `NodePool` e l'esempio`workload`, esamina quanto segue e la definizione del Pod NodePool e salva con nome e: `nodepool-gpu.yaml` `pod.yaml`

nodepool-gpu.yaml

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
```

```
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      terminationGracePeriod: 24h0m0s
```

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
    - "nvidia-smi"
    resources:
      requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

Il selettore `eks.amazonaws.com/compute-type: auto` richiede che il carico di lavoro sia implementato su un nodo della modalità automatica di Amazon EKS. NodePool Inoltre, imposta un limite che consente di programmare solo pod con tolleranze per le GPU NVIDIA.

Applica il carico di lavoro NodePool and al tuo cluster.

```
kubectl apply -f nodepool-gpu.yaml
kubectl apply -f pod.yaml
```

Verrà visualizzato l'output seguente:

```
nodepool.karpenter.sh/gpu created
pod/nvidia-smi created
```

Attendi qualche secondo e controlla i nodi del cluster. Ora dovresti vedere un nuovo nodo fornito nel cluster della modalità automatica di Amazon EKS:

```
> kubectl get nodes

NAME        TYPE           CAPACITY    ZONE         NODE                   READY   AGE
gpu-dnknr   g6e.2xlarge    on-demand   us-west-2b   i-02315c7d7643cdee6   True    76s
```

## Fase 2: Convalida

Puoi vedere Amazon EKS Auto Mode avviato a `g6e.2xlarge` anziché a `g6.2xlarge` poiché il carico di lavoro richiedeva un'istanza con un L40SGPU, in base ai seguenti vincoli di pianificazione Kubernetes:

```
...
  nodeSelector:
    eks.amazonaws.com/instance-gpu-name: l40s
...
    requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
```

Ora, esamina i log del contenitore eseguendo il seguente comando:

```
kubectl logs nvidia-smi
```

Output di esempio:

```
+-----------------------------------------------------------------------------
+
| NVIDIA-SMI 535.230.02             Driver Version: 535.230.02   CUDA Version: 12.2
 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA L40S                    On  | 00000000:30:00.0 Off |                    0
 |
| N/A   27C    P8              23W / 350W |      0MiB / 46068MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------+----------------------
+----------------------+

+-----------------------------------------------------------------------------
+
| Processes:
 |
|  GPU   GI   CI        PID   Type   Process name                         GPU Memory
 |
|        ID   ID                                                          Usage
 |
|
=============================================================================|
|  No running processes found
 |
+-----------------------------------------------------------------------------
+
```

Puoi vedere che il container ha rilevato che è in esecuzione su un'istanza con una GPU `NVIDIA` e che non hai dovuto installare alcun driver di dispositivo, poiché questa è gestita dalla modalità automatica di Amazon EKS.

## Fase 3: Clean-up

Per rimuovere tutti gli oggetti creati, usa `kubectl` per eliminare la distribuzione di esempio e NodePool quindi il nodo viene terminato:

```
kubectl delete -f nodepool-gpu.yaml
kubectl delete -f pod.yaml
```

## Esempio di riferimento NodePools

### Crea una NVIDIA NodePool

Quanto segue NodePool definisce:

- Solo istanze di avvio della famiglia `g6e` e `g6`
- Consolidare i nodi quando sono vuoti per 1 ora
  - Il valore di 1 ora per `consolidateAfter` supporta carichi di lavoro intensi e riduce l'abbandono dei nodi. Puoi eseguire l'ottimizzazione di `consolidateAfter` in base ai requisiti del carico di lavoro.

Esempio NodePool con famiglia di istanze GPU e consolidamento

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
```

```
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      terminationGracePeriod: 24h0m0s
```

Invece di impostare il, `eks.amazonaws.com/instance-gpu-name` potresti utilizzare `eks.amazonaws.com/instance-family` per specificare la famiglia di istanze. Per altre etichette note che influenzano la revisione della pianificazione, consultare the section called "Etichette supportate dalla modalità automatica di EKS".

Se hai requisiti di archiviazione specifici, puoi ottimizzare lo storage `iops` temporaneo dei nodi `size` e `throughput` crearne uno personalizzato NodeClass a cui fare riferimento nel. NodePool Scopri di più sulle opzioni configurabili NodeClass .

Esempio di configurazione di archiviazione per NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  ephemeralStorage:
    iops: 3000
    size: 80Gi
    throughput: 125
```

Definire un AWS Trainium e AWS Inferentia NodePool

Quanto segue NodePool contiene un `eks.amazonaws.com/instance-category` set che dice, avvia solo le istanze della famiglia Inferentia e Trainium:

```
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values:
            - inf
```

```
            - trn
```

# Configurazione delle impostazioni della modalità automatica EKS

Questo capitolo descrive come configurare aspetti specifici dei cluster in modalità automatica Amazon Elastic Kubernetes Service (EKS). Sebbene la modalità automatica EKS gestisca automaticamente la maggior parte dei componenti dell'infrastruttura, è possibile personalizzare alcune funzionalità per soddisfare i requisiti del tuo carico di lavoro.

Utilizzando le opzioni di configurazione descritte in questo argomento, è possibile modificare le impostazioni di rete, le risorse di calcolo e i comportamenti di bilanciamento del carico, mantenendo al contempo i vantaggi della gestione automatizzata dell'infrastruttura. Prima di apportare modifiche alla configurazione, esamina le opzioni disponibili nelle sezioni seguenti per determinare l'approccio più adatto alle tue esigenze.

| Quali funzionalità vuoi configurare? | Opzione di configurazione |
| --- | --- |
| Rete e archiviazione dei nodi<br>• Configura il posizionamento dei nodi tra sottoreti pubbliche e private<br>• Definisci gruppi di sicurezza personalizzati per il controllo degli accessi ai nodi<br>• Personalizza le policy di traslazione degli indirizzi di rete (SNAT)<br>• Abilita le policy di rete Kubernetes, la registrazione e il monitoraggio dettagliati delle policy di rete dettagliate<br>• Imposta parametri di archiviazione temporanei (dimensioni, IOPS, throughput)<br>• Configura l'archiviazione temporanea crittografata con chiavi KMS personalizzate<br>• Isola il traffico dei pod in sottoreti separate dai nodi | the section called "Creazione di una classe di nodi" |
| Risorse di calcolo dei nodi | the section called "Creazione di pool di nodi" |

| Quali funzionalità vuoi configurare? | Opzione di configurazione |
| --- | --- |
| • Seleziona tipi e famiglie di istanze EC2 specifici<br>• Definisci le architetture della CPU (x86_64, ARM64)<br>• Configurazione dei tipi di capacità (On-Demand, Spot)<br>• Specifica zone di disponibilità<br>• Configura i taint e le etichette dei nodi<br>• Imposta limiti di risorse per l'utilizzo di CPU e memoria | |
| Static-capacity pool di nodi<br>• Mantieni un numero fisso di nodi indipendentemente dalla richiesta di pod<br>• Configura la capacità prevedibile per le istanze riservate<br>• Imposta un numero di repliche specifico per un'impronta coerente dell'infrastruttura<br>• Scala manualmente i pool di nodi statici<br>• Monitora lo stato del pool di nodi con capacità statica | the section called "Crea un pool di nodi statico" |

| Quali funzionalità vuoi configurare? | Opzione di configurazione |
| --- | --- |
| Impostazioni Application Load Balancer<br>• Implementa bilanciatori del carico interni o connessi a Internet<br>• Configura il load balancer tra zone<br>• Imposta periodi di timeout di inattività<br>• Abilita HTTP/2 e supporta WebSocket<br>• Configura i parametri del controllo dell'integrità<br>• Specifica le impostazioni del certificato TLS<br>• Definisci gli attributi dei gruppi di destinazione<br>• Imposta il tipo di indirizzo IP (IPv4, dual-stack) | the section called "Creare una classe in ingresso" |
| Impostazioni Network Load Balancer<br>• Configura il routing IP diretto del pod<br>• Abilita il load balancer tra zone<br>• Imposta un timeout di inattività della connessione<br>• Configura i parametri del controllo dell'integrità<br>• Specifica il posizionamento della sottorete<br>• Imposta il tipo di indirizzo IP (IPv4, dual-stack)<br>• Configura l'IP di conservazione dell'origine del client<br>• Definisci gli attributi dei gruppi di destinazione | the section called "Creare servizio" |

| Quali funzionalità vuoi configurare? | Opzione di configurazione |
| --- | --- |
| Gruppi target composti da più cluster<br>• Scopri come il controller di bilanciamento del carico gestisce l'appartenenza al gruppo target<br>• Abilita MultiCluster il supporto per condivide re un gruppo target su più cluster Kubernetes<br>• Supporta obiettivi arbitrari provenienti da altre fonti in un gruppo target condiviso | the section called "Multi-cluster gruppi target" |
| Estensione dell'RBAC per i controller gestiti<br>• Scopri come i controller gestiti EKS Auto Mode si autenticano nel tuo cluster<br>• Concedi un RBAC Kubernetes aggiuntivo ai controller gestiti tramite il gruppo `eks:manag ed`<br>• Definisci la concessione con/e con il privilegi o minimo `Role` `RoleBinding` `resourceN ames` | the section called "Concedi RBAC ai controller gestiti" |
| Impostazioni della classe di archiviazione<br>• Definisci i tipi di volume EBS (gp3, io1, io2, eccetera)<br>• Configura la crittografia dei volumi e l'utilizzo delle chiavi KMS<br>• Imposta i parametri IOPS e di throughput<br>• Imposta come classe di archiviazione predefinita<br>• Definisci tag personalizzati per i volumi assegnati | the section called "Crea StorageClass" |

| Quali funzionalità vuoi configurare? | Opzione di configurazione |
| --- | --- |
| Controlla l'utilizzo dell'ODCR<br>• Configurare la distribuzione del carico di lavoro in EC2 Capacity Reservations On-Demand<br>• Seleziona in modo esplicito ODCR specifici in base all'ID per un utilizzo mirato della capacità<br>• Utilizza la selezione basata su tag per indirizzare gruppi di ODCR<br>• Filtra gli ODCR in base alla proprietà dell'account per scenari con più account AWS<br>• Controlla se i carichi di lavoro utilizzano automaticamente gli ODCR aperti | the section called "Controllare le prenotazioni della capacità" |
| Sicurezza avanzata del nodo<br>• Abilita i moduli crittografici convalidati FIPS 140-2<br>• Configurare la modalità di sicurezza del blocco del kernel | the section called "Sicurezza avanzata" |

| Quali funzionalità vuoi configurare? | Opzione di configurazione |
| --- | --- |
| Ottimizzazione dei costi<br>• Scopri come il consolidamento, il bin-packing e il corretto dimensionamento riducono i costi di elaborazione<br>• Identifica le configurazioni che bloccano il consolidamento (PDB restrittivi) `do-not-disrupt`<br>• Imposta come limite di costo NodePool `limits`<br>• Limita le famiglie e le dimensioni delle istanze per controllare i costi complessivi<br>• Applica i modelli consigliati per carichi di lavoro CI/CD intensivi | Ottimizzazione dei costi in modalità EKS Auto |

# Creazione di una classe di nodi per Amazon EKS

Le classi di nodi Amazon EKS sono modelli che forniscono un controllo granulare sulla configurazione dei nodi gestiti dalla modalità automatica di EKS. Una classe di nodi definisce le impostazioni a livello di infrastruttura che si applicano ai gruppi di nodi del cluster EKS, tra cui la configurazione di rete, le impostazioni di storage e il tagging delle risorse. Questo argomento mostra come creare e configurare una classe di nodi per soddisfare i tuoi requisiti operativi specifici.

Quando è necessario personalizzare le modalità di provisioning e configurazione delle istanze EC2 da parte della modalità automatica di EKS oltre le impostazioni predefinite, la creazione di una classe di nodi fornisce un controllo preciso sui parametri critici dell'infrastruttura. Ad esempio, è possibile specificare il posizionamento di sottoreti private per una maggiore sicurezza, configurare lo storage temporaneo delle istanze per carichi di lavoro sensibili alle prestazioni o applicare tag personalizzati per l'allocazione dei costi.

## Creazione di una classe di nodi

Per creare una `NodeClass`, segui questi passaggi:

1. Crea un file YAML (ad esempio, `nodeclass.yaml`) con la configurazione della classe di nodi

2. Applica la configurazione al cluster utilizzando `kubectl`
3. Fai riferimento alla classe di nodi nella configurazione del tuo pool di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

`kubectl` deve essere installato e configurato. Per ulteriori informazioni, consulta Configurazione.

Esempio di classe di nodi di base

Ecco un esempio di classe di nodi:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: private-compute
spec:
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
  ephemeralStorage:
    size: "160Gi"
```

Ciò NodeClass aumenta la quantità di spazio di archiviazione temporaneo sul nodo.

Applica questa configurazione usando:

```
kubectl apply -f nodeclass.yaml
```

Poi fai riferimento alla classe di nodi nella configurazione del pool di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

## Creazione di una voce di accesso alla classe di nodi

Se crei una classe di nodi personalizzata, devi creare una voce di accesso EKS per consentire ai nodi di unirsi al cluster. Quando si utilizzano la classe di nodi e i pool di nodi integrati, EKS crea automaticamente le voci di accesso.

Per informazioni sul funzionamento delle voci di accesso, consulta the section called "Voci di accesso".

Quando si creano voci di accesso per le classi di nodi della modalità automatica di EKS, è necessario usare il tipo di voce di accesso `EC2`.

Creazione di un accesso con CLI

Per creare una voce di accesso per i nodi EC2 e associare la policy dei nodi di EKS Auto:

Aggiorna i comandi CLI seguenti con il nome del cluster e l'ARN del ruolo del nodo. Il ruolo del nodo ARN viene specificato nella classe di nodi YAML.

```
# Create the access entry for EC2 nodes
aws eks create-access-entry \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --type EC2

# Associate the auto node policy
aws eks associate-access-policy \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy \
  --access-scope type=cluster
```

Crea una voce di accesso con CloudFormation

Per creare una voce di accesso per i nodi EC2 e associare la policy dei nodi di EKS Auto:

Aggiorna quanto segue CloudFormation con il nome del cluster e l'ARN del ruolo del nodo. Il ruolo del nodo ARN viene specificato nella classe di nodi YAML.

```
EKSAutoNodeRoleAccessEntry:
  Type: AWS::EKS::AccessEntry
  Properties:
    ClusterName: <cluster-name>
    PrincipalArn: <node-role-arn>
    Type: "EC2"
    AccessPolicies:
      - AccessScope:
          Type: cluster
        PolicyArn: arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy
```

```
  DependsOn: [ <cluster-name> ] # previously defined in CloudFormation
```

Per informazioni sulla distribuzione degli CloudFormation stack, vedi Guida introduttiva con CloudFormation

## Specifiche della classe di nodi

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Required fields

  # role and instanceProfile are mutually exclusive fields.
  role: MyNodeRole  # IAM role for EC2 instances
  # instanceProfile: eks-MyNodeInstanceProfile  # IAM instance-profile for EC2
 instances

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0123456789abcdef0"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
    # Alternative approaches:
    # - id: "sg-0123456789abcdef0"
    # - name: "eks-cluster-security-group"

  # Optional: Pod subnet selector for advanced networking
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0987654321fedcba0"
  # must include Pod security group selector also
  podSecurityGroupSelectorTerms:
    - tags:
```

```
        Name: "eks-pod-sg"
    # Alternative using direct security group ID
    # - id: "sg-0123456789abcdef0"

  # Optional: Selects on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        Name: "targeted-odcr"
      # Optional owning account ID filter
      owner: "012345678901"

  # Optional fields
  snatPolicy: Random  # or Disabled

  networkPolicy: DefaultAllow  # or DefaultDeny
  networkPolicyEventLogs: Disabled  # or Enabled

  ephemeralStorage:
    size: "80Gi"    # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000      # Range: 3000-16000
    throughput: 125 # Range: 125-1000
    # Optional KMS key for encryption
    kmsKeyID: "arn:aws: kms:region:account:key/key-id"
    # Accepted formats:
    # KMS Key ID
    # KMS Key ARN
    # Key Alias Name
    # Key Alias ARN

  advancedNetworking:
    # Optional: Controls whether public IP addresses are assigned to instances that are
 launched with the nodeclass.
    # If not set, defaults to the MapPublicIpOnLaunch setting on the subnet.
    associatePublicIPAddress: false

    # Optional: Forward proxy, commonly requires certificateBundles as well
    # for EC2, see https://repost.aws/knowledge-center/eks-http-proxy-containerd-
automation
    httpsProxy: http://192.0.2.4:3128 #commonly port 3128 (Squid) or 8080 (NGINX) #Max
 255 characters
    #httpsProxy: http://[2001:db8::4]:3128 # IPv6 address with port, use []
```

```
    noProxy: #Max 50 entries
        - localhost #Max 255 characters each
        - 127.0.0.1
        #- ::1 # IPv6 localhost
        #- 0:0:0:0:0:0:0:1 # IPv6 localhost
        - 169.254.169.254 # EC2 Instance Metadata Service
        #- [fd00:ec2::254] # IPv6 EC2 Instance Metadata Service
        # Domains to exclude, put all VPC endpoints here
        - .internal
        - .eks.amazonaws.com
    # ipv4PrefixSize is default to Auto which is prefix and fallback to secondary IP.
"32" is the secondary IP mode.
    ipv4PrefixSize: Auto # or "32"

    # enableV4Egress is default to true. Setting it to false when using network policy
or blocking IPv4 traffic in IPv6 clusters
    enableV4Egress: false

    # Optional: Static network interface configuration for EFA workloads
    networkInterfaces:
      # The primary ENI (networkCardIndex 0, deviceIndex 0) must use interfaceType:
interface
      # secondaryIPv4Count and secondaryIPv4PrefixCount are only supported on
networkCardIndex 0
    - deviceIndex: 0
      interfaceType: interface
      networkCardIndex: 0
      secondaryIPv4Count: 10
    - deviceIndex: 1
      interfaceType: interface
      networkCardIndex: 0
      secondaryIPv4PrefixCount: 2 # Only one of secondaryIPv4Count or
secondaryIPv4PrefixCount per ENI
    - deviceIndex: 0
      interfaceType: efa-only
      networkCardIndex: 1

  advancedSecurity:
    # Optional, US regions only: Specifying `fips: true` will cause nodes in the
nodeclass to run FIPS compatible AMIs.
    fips: false

  # Optional: Custom certificate bundles.
  certificateBundles:
```

```
    - name: "custom-cert"
      data: "base64-encoded-cert-data"

  # Optional: EC2 Placement Group
  placementGroupSelector:
    name: "targeted-pg"
    # Alternative use direct placement group ID
    # id: "pg-02465754522cda020"

  # Optional: Additional EC2 tags (with restrictions)
  tags:
    Environment: "production"
    Team: "platform"
    # Note: Cannot use restricted tags like:
    # - kubernetes.io/cluster/*
    # - karpenter.sh/provisioner-name
    # - karpenter.sh/nodepool
    # - karpenter.sh/nodeclaim
    # - karpenter.sh/managed-by
    # - eks.amazonaws.com/nodeclass
```

## Considerazioni

- Se vuoi verificare la quantità di storage locale di cui dispone un'istanza, puoi descrivere il nodo per visualizzare la risorsa di storage temporaneo.
- Crittografia dei volumi: EKS utilizza la chiave KMS personalizzata configurata per crittografare il volume principale di sola lettura dell'istanza e il volume di dati. read/write
- Sostituisci il ruolo IAM del nodo: se modifichi il ruolo IAM del nodo associato a `NodeClass`, dovrai creare una nuova voce d'accesso. EKS crea automaticamente una voce d'accesso per il ruolo IAM del nodo durante la creazione del cluster. Il ruolo IAM del nodo richiede la policy di accesso di `AmazonEKSAutoNodePolicy` EKS. Per ulteriori informazioni, consulta the section called "Voci di accesso".
- densità massima di pod: EKS limita il numero massimo di pod su un nodo a 110. Questo limite viene applicato dopo il calcolo del numero massimo di pod attuale. Per ulteriori informazioni, consulta the section called "Tipi di istanza Amazon EC2".
- Tag: se vuoi propagare i tag da Kubernetes a EC2, devi configurare autorizzazioni IAM aggiuntive. Per ulteriori informazioni, consulta the section called "Identità e accesso".
- Classe di nodi predefinita: non assegnare un nome alla classe di nodi `default` personalizzata. Questo perché la modalità automatica di EKS include una `NodeClass` denominata `default` che

viene fornita automaticamente quando si abilita almeno un `NodePool` integrato. Per informazioni sull'abilitazione dei `NodePools`, consulta the section called "Riesamina i pool di nodi integrati".

- comportamento di **`subnetSelectorTerms`** con più sottoreti: se esistono più sottoreti che soddisfano le condizioni di `subnetSelectorTerms` o che vengono fornite tramite ID, la modalità automatica di EKS crea nodi distribuiti tra le sottoreti.
  - Se le sottoreti si trovano in zone di disponibilità (AZ) diverse, è possibile usare funzionalità di Kubernetes come i vincoli di distribuzione della topologia dei pod e il routing basato sulla topologia per distribuire rispettivamente i pod e il traffico tra le zone.
  - Se nella stessa AZ sono presenti più sottoreti che corrispondono al `subnetSelectorTerms`, la modalità automatica di EKS crea pod su ciascun nodo distribuito tra le sottoreti in quella AZ. La modalità automatica di EKS crea interfacce di rete secondarie su ogni nodo nelle altre sottoreti della stessa AZ. Decide in base al numero di indirizzi IP disponibili in ciascuna sottorete, per usare le sottoreti in modo più efficiente. Non è tuttavia possibile specificare quale sottorete viene usata dalla modalità automatica di EKS per ogni Pod; se hai bisogno che i Pod funzionino in sottoreti specifiche, usa the section called "Sottoreti e gruppi di sicurezza separati per i Pods".
- Restrizioni relative alla strategia dei gruppi di collocamento: ogni strategia di gruppo di collocamento (cluster, partizione, spread) presenta limitazioni specifiche sui tipi di istanze, sugli AZ e sulla capacità. Per i dettagli, consulta le strategie dei gruppi di collocamento nella Amazon EC2 User Guide.
- Gruppo di spread placement — limite di 7 istanze: un gruppo di spread placement a livello di rack consente un massimo di 7 istanze in esecuzione per zona di disponibilità per gruppo. Ciò crea i seguenti casi limite:
  - Sostituzione Drift bloccata al massimo della capacità: EKS Auto Mode avvia un nodo sostitutivo prima di terminare quello precedente. Quando un PG diffuso ha 7 istanze in un AZ, l'avvio sostitutivo fallisce e il nodo drifted rimane in funzione fino a quando non si libera uno slot.
  - Tutti gli AZ al massimo della loro capacità: se ogni AZ nello spread PG è al limite di 7 istanze, non è possibile programmare alcuna sostituzione. I nodi Drifted o candidati al consolidamento rimangono attivi a tempo indeterminato.
  - Nessun fallback all'esterno del gruppo di posizionamento: EKS Auto Mode non tenta di avviare istanze sostitutive all'esterno del gruppo di posizionamento.
  - Soluzione alternativa: utilizzare la politica di `WhenEmpty` consolidamento (). `consolidationPolicy: WhenEmpty` I nodi vengono eliminati solo dopo che tutti i pod non appartenenti a daemonset si sono esauriti, liberando uno slot PG senza dover prima avviare un altro server sostitutivo. Nota che drift utilizza sempre replace-then-delete indipendentemente dalla politica di consolidamento, quindi drift rimane bloccato al massimo della capacità.

- Blocco AZ del gruppo di posizionamento del cluster: una volta che la prima istanza viene avviata in un gruppo di posizionamento del cluster, il PG viene aggiunto a quell'AZ. Se si NodePool consentono più AZ, i lanci paralleli durante lo scale-up iniziale potrebbero avere esito negativo: uno ha esito positivo e blocca l'AZ, gli altri falliranno a causa di errori di capacità. Inserite la AZ nelle vostre NodePool esigenze per evitare guasti transitori.
- Gruppo di posizionamento delle partizioni: i gruppi di posizionamento delle partizioni sono supportati senza vincoli aggiuntivi oltre ai limiti standard EC2.
- Il consolidamento può spostare i pod fuori da un gruppo di collocamento: se un pod non ha vincoli di pianificazione relativi al gruppo di posizionamento (ad esempio attivo`eks.amazonaws.com/placement-group-id`), il consolidamento può spostarlo `nodeSelector` in un nodo esterno al PG. Le applicazioni che richiedono l'appartenenza a un gruppo di collocamento devono esprimerlo tramite vincoli a livello di pod.
- Gruppo di posizionamento inesistente o eliminato: se un `NodeClass` fa riferimento a un gruppo di posizionamento che non esiste o è stato eliminato, non viene avviata alcuna istanza. Il formato ID del gruppo di collocamento viene convalidato al momento dell'ammissione, ma l'esistenza viene verificata solo al momento dell'avvio. Se un gruppo di posizionamento viene eliminato mentre i nodi sono in esecuzione, i nodi esistenti vengono contrassegnati come drifted e rimangono in esecuzione a tempo indeterminato perché anche i lanci di Drift Replacement sono bloccati.
- Hugepages: per informazioni sulla configurazione di hugepages sui nodi, consulta. the section called “Configura enormi pagine sui nodi”

## Sottoreti e gruppi di sicurezza separati per i Pods

I `podSecurityGroupSelectorTerms` campi `podSubnetSelectorTerms` and consentono configurazioni di rete avanzate consentendo ai Pods di utilizzare sottoreti e gruppi di sicurezza diversi rispetto ai rispettivi nodi. Entrambi i campi devono essere specificati insieme. Questa separazione offre maggiore controllo sul routing del traffico di rete e sulle policy di sicurezza.

> **Note**
>
> Questa funzionalità è diversa dalla funzionalità Security Groups for Pods (SGPP) utilizzata con il VPC CNI per il calcolo in modalità automatica non EKS. SGPP non è supportato nella modalità EKS Auto. Utilizza invece `podSecurityGroupSelectorTerms` in `NodeClass` per applicare gruppi di sicurezza separati al traffico Pod. I gruppi di sicurezza si applicano

a `NodeClass` livello, il che significa che tutti i Pod sui nodi che utilizzano `NodeClass` condividono gli stessi gruppi di sicurezza Pod.

### Come funziona

Quando configuri `podSubnetSelectorTerms` e`podSecurityGroupSelectorTerms`:

1. L'ENI primario del nodo utilizza le sottoreti e i gruppi di sicurezza di e. `subnetSelectorTerms` `securityGroupSelectorTerms` A questa interfaccia viene assegnato solo l'indirizzo IP del nodo.
2. EKS Auto Mode crea ENI secondari nelle sottoreti corrispondenti`podSubnetSelectorTerms`, con i gruppi di sicurezza collegati. `podSecurityGroupSelectorTerms` Gli indirizzi IP dei pod vengono allocati da questi ENI secondari utilizzando i prefissi /28 per impostazione predefinita, con riserva automatica agli IP secondari (/32) quando non è disponibile un blocco di prefisso contiguo. Se `ipv4PrefixSize` è impostato su in, vengono utilizzati solo gli IP secondari. `"32"` `advancedNetworking`
3. I gruppi di sicurezza specificati in `podSecurityGroupSelectorTerms` si applicano al traffico Pod all'interno del VPC. Per il traffico destinato all'esterno del VPC, i Pods utilizzano l'ENI primario del nodo (e i relativi gruppi di sicurezza) perché la traduzione degli indirizzi di rete di origine (SNAT) traduce l'IP del Pod nell'IP del nodo. Puoi modificare questo comportamento con il campo in. `snatPolicy` `NodeClass`

### Casi d'uso

Usa `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` quando hai bisogno di:

- Applica diversi gruppi di sicurezza per controllare separatamente il traffico per nodi e pod.
- Separa il traffico dell'infrastruttura (comunicazione da nodo a nodo) dal traffico delle applicazioni (comunicazione). Pod-to-Pod
- Applicare configurazioni di rete diverse alle sottoreti dei nodi rispetto alle sottoreti Pod.
- Configurare i proxy inversi o il filtraggio di rete in modo specifico per il traffico dei nodi senza influire sul traffico dei Pod. Utilizza `advancedNetworking` e `certificateBundles` per definire il tuo proxy inverso e qualsiasi certificato autofirmato o privato per il proxy.

Configurazione di esempio

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  # Subnets and security groups for EC2 instances (nodes)
  subnetSelectorTerms:
    - tags:
        Name: "node-subnet"
        kubernetes.io/role/internal-elb: "1"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  # Separate subnets and security groups for Pods
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"

  podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
```

Considerazioni per sottoreti Pod e gruppi di sicurezza separati

- Ambito del gruppo di sicurezza: i gruppi di sicurezza di `podSecurityGroupSelectorTerms` sono collegati agli ENI secondari e si applicano al traffico Pod all'interno del VPC. Quando SNAT è abilitato (impostazione predefinita`snatPolicy: Random`), il traffico in uscita dal VPC viene tradotto nell'indirizzo IP ENI primario del nodo, quindi i gruppi di sicurezza del nodo `securityGroupSelectorTerms` si applicano invece a quel traffico. Se lo impostisnatPolicy: `Disabled`, i Pods utilizzano i propri indirizzi IP per tutto il traffico e devi assicurarti che i gruppi di routing e sicurezza siano configurati di conseguenza.
- NodeClass-level granularità: i gruppi di sicurezza dei Pod si applicano a tutti i Pod pianificati sui nodi utilizzando il. `NodeClass` Per applicare diversi gruppi di sicurezza a diversi carichi di lavoro,

crea `NodePool` risorse binarie separate `NodeClass` e utilizza taint, tolleranze o selettori di nodi per pianificare i carichi di lavoro sui nodi appropriati.

- Densità di pod ridotta: è possibile eseguire un numero inferiore di pod su ogni nodo perché l'interfaccia di rete principale del nodo è riservata all'IP del nodo e non può essere utilizzata per i pod.
- Limitazioni del selettore di sottorete: lo standard `subnetSelectorTerms` e le `securityGroupSelectorTerms` configurazioni non si applicano alla selezione della sottorete o del gruppo di sicurezza di Pod.
- Pianificazione della rete: assicurati che lo spazio degli indirizzi IP sia adeguato sia nella sottorete del nodo che in quella del pod per supportare i requisiti del tuo carico di lavoro.
- Configurazione del routing: verifica che la tabella di routing e l'Access Control List (ACL) di rete delle sottoreti Pod siano configurati correttamente per la comunicazione tra nodo e sottoreti Pod.
- Zone di disponibilità: verifica di aver creato sottoreti del pod su più zone di disponibilità. Se si utilizza una sottorete Pod specifica, deve trovarsi nella stessa AZ della sottorete del nodo AZ.

## Modalità IP secondaria per pod

Il `ipv4PrefixSize` campo consente configurazioni di rete avanzate assegnando solo indirizzi IP secondari ai nodi. Questa funzionalità non assegna prefissi (/28) ai nodi e mantiene solo un IP secondario come minimalIPTarget.

### Casi d'uso

Usa `ipv4PrefixSize` quando devi:

- Utilizzo IP ridotto: verrà riscaldato un solo indirizzo IP in ogni nodo.
- Basso tasso di abbandono dei pod: la velocità di creazione dei pod non è una delle principali preoccupazioni.
- Nessuna frammentazione del prefisso: Prefix-caused la frammentazione è un problema o un ostacolo importante all'uso della modalità automatica.

### Configurazione di esempio

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
```

```
spec:
  role: MyNodeRole

  advancedNetworking:
    ipv4PrefixSize: "32"
```

Considerazioni sulla modalità IP secondaria

- Velocità di creazione dei pod ridotta: poiché viene riscaldato solo un IP secondario, il servizio IPAM richiede più tempo per eseguire il provisioning degli IP quando vengono creati più pod.

## Disabilita l'uscita IPv4 dai pod IPv6 nei cluster IPv6.

Il `enableV4Egress` `true` campo è predefinito. Per i cluster IPv6 in modalità automatica, la funzionalità può essere disabilitata in modo che la modalità automatica non crei un'interfaccia IPv4 di sola uscita per i pod IPv6. Questo è importante perché l'interfaccia di uscita IPv4 non è soggetta all'applicazione dei criteri di rete. Le politiche di rete vengono applicate solo sull'interfaccia principale del Pod (eth0).

Casi d'uso

Usa `enableV4Egress` quando devi:

- Usa il cluster IPv6: il traffico in uscita IPv4 è consentito per impostazione predefinita.
- Usa la politica di rete: attualmente EKS Network Policy non supporta il dual stack. La disabilitazione `enableV4Egress` può impedire la fuoriuscita inaspettata del traffico del pod su IPv4.

Configurazione di esempio

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  advancedNetworking:
    enableV4Egress: false
```

Considerazioni per la disabilitazione di EnableV4egress

- Politica di rete nel cluster IPv6: i cluster IPv6 consentono il traffico IPv4 per impostazione predefinita. L'impostazione `enableV4Egress: false` blocca il traffico in uscita IPv4, fornendo una maggiore sicurezza soprattutto se utilizzata con i criteri di rete.

## Configurazione statica dell'interfaccia di rete

Il `networkInterfaces` campo sotto `advancedNetworking` consente di definire staticamente le interfacce di rete collegate alle istanze all'avvio. Viene utilizzato principalmente per configurare i dispositivi Elastic Fabric Adapter (EFA) per comunicazioni tra nodi ad alte prestazioni nei carichi di lavoro di formazione e inferenza distribuiti. Questa funzionalità può essere associata a pool di nodi con capacità statica per mantenere i nodi preriscaldati. EFA-ready Per ulteriori informazioni sulla gestione dei dispositivi EFA in EKS, vedere. the section called "Dispositivi EFA" Per la configurazione EFA consigliata per tipi di istanze specifici, consulta Massimizzare la larghezza di banda di rete per i tipi di EFA-enabled istanza nella Amazon EC2 User Guide.

Come funziona

Ogni voce in `networkInterfaces` definisce un'interfaccia di rete collegata all'istanza durante l'avvio. Ogni voce specifica una `networkCardIndex` (la scheda di rete, dove `0` è la principale), una `deviceIndex` (la posizione del dispositivo su quella scheda) e una `interfaceType` (`interface`per il IP-based traffico standard o `efa-only` per le interfacce EFA dedicate al traffico RDMA senza indirizzi IP). L'ENI primario (`networkCardIndex: 0`,`deviceIndex: 0`) deve essere utilizzato `interfaceType: interface` per supportare la comunicazione tra i nodi. IP-based Le `efa-only` interfacce supportano solo le funzionalità dei dispositivi EFA per RDMA e non possono essere configurate con indirizzi IP.

È possibile assegnare la capacità IP alle interfacce sulla scheda di rete primaria (`networkCardIndex: 0`) utilizzando `secondaryIPv4Count` (ogni unità fornisce 1 indirizzo IP) o `secondaryIPv4PrefixCount` (ogni unità fornisce un prefisso /28 con 16 indirizzi IP). È possibile utilizzarne solo uno per interfaccia e sono supportati solo su. `networkCardIndex: 0` Non possono essere utilizzati sulle `efa-only` interfacce.

> **Important**
>
> Quando `networkInterfaces` è configurato, EKS Auto Mode non allega IP, prefissi o ENI aggiuntivi dopo l'avvio dell'istanza. Solo le interfacce e gli indirizzi IP configurati all'avvio

sono disponibili per Pods. È necessario pianificare la densità dei pod in base al numero di IP configurati.

Considerazioni

L'IPv6 non è supportato con interfacce di rete definite staticamente. `associatePublicIPAddress`non è compatibile quando è definita più di un'interfaccia di rete, quindi i nodi che utilizzano più interfacce non possono avere IP pubblici (sia in cluster solo pubblici che in cluster misti). public/private

Quando si utilizzano `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` con interfacce di rete statiche, configura l'ENI primario con 0 IP secondari. L'ENI primario utilizza la sottorete del nodo e i gruppi di sicurezza, mentre il traffico Pod utilizza la configurazione della sottorete pod sugli ENI secondari.

Esempio: EFA-only interfacce per l'addestramento delle GPU

L'esempio seguente mostra una `NodeClass` configurazione con EFA-only interfacce per un'istanza GPU utilizzata nell'addestramento distribuito. L'interfaccia principale ha un prefisso /28 (16 IP pod) e 4 EFA-only interfacce aggiuntive forniscono la connettività RDMA. Questa configurazione può essere associata a un pool di nodi con capacità statica per mantenere i nodi preriscaldati. EFA-ready

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: efa-training
spec:
  role: MyNodeRole
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
  securityGroupSelectorTerms:
    - tags:
        Name: "efa-security-group"
  placementGroupSelector:
    name: "ml-training-pg"
  advancedNetworking:
    networkInterfaces:
    - deviceIndex: 0
      interfaceType: interface
```

```
      networkCardIndex: 0
      secondaryIPv4PrefixCount: 1
    - deviceIndex: 1
      interfaceType: efa-only
      networkCardIndex: 0
    - deviceIndex: 0
      interfaceType: efa-only
      networkCardIndex: 1
    - deviceIndex: 0
      interfaceType: efa-only
      networkCardIndex: 2
    - deviceIndex: 0
      interfaceType: efa-only
      networkCardIndex: 3
```

## Configura enormi pagine sui nodi

Usa il `advancedCompute.hugepages` campo per preallocare le hugepages sui nodi in. `NodeClass` Hugepages migliora le prestazioni di accesso alla memoria per carichi di lavoro sensibili alla latenza come l'elaborazione ad alte prestazioni (HPC), i database e le applicazioni ad alta intensità di rete. Per utilizzare hugepage, configura i tuoi Pod per richiedere esplicitamente le risorse di hugepage. Per ulteriori informazioni, consulta Manage nella documentazione di Kubernetes. HugePages

### Configurazione di esempio

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: hugepages-compute
spec:
  role: MyNodeRole

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  advancedCompute:
```

```
    hugepages:
      pages:
        - size: "2Mi"
          count: 512
```

Considerazioni su Hugepages

Considera quanto segue quando configuri hugepages su un: NodeClass

- Prenotazione della memoria: le pagine enormi vengono conteggiate nella memoria totale dell'istanza. Amazon EKS contrassegna i tipi di istanza come incompatibili con NodeClass quando la prenotazione di hugepages supera l'80% della memoria totale.
- Mutua esclusione EFA: EFA e hugepages non possono coesistere sullo stesso. NodeClass Se imposti, la convalida fallisce. `interfaceType: efa-only` NodeClass Amazon EKS non fornisce pod che richiedono pagine `vpc.amazonaws.com/efa` enormi. NodeClass Usane uno separato NodeClass per i carichi di lavoro EFA.

## Crea un pool di nodi per la modalità automatica di EKS

I pool di nodi Amazon EKS forniscono un modo flessibile per gestire le risorse di calcolo nel cluster Kubernetes. Questo argomento mostra come creare e configurare pool di nodi usando Karpenter, uno strumento di provisioning dei nodi che aiuta a ottimizzare il dimensionamento dei cluster e l'utilizzo delle risorse. Con la NodePool risorsa di Karpenter, puoi definire requisiti specifici per le tue risorse di calcolo, inclusi tipi di istanze, zone di disponibilità, architetture e tipi di capacità.

Non è possibile modificare il `system` integrato e i pool di nodi `general-purpose`. È solo possibile abilitarli o disabilitarli. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

Le NodePool specifiche consentono un controllo granulare sulle risorse di calcolo del cluster EKS attraverso varie etichette e requisiti supportati. Queste includono opzioni per specificare le categorie di istanze EC2, le configurazioni della CPU, le zone di disponibilità, le architetture (ARM64/AMD64) e i tipi di capacità (spot o on-demand). È inoltre possibile impostare limiti di risorse per l'utilizzo della CPU e della memoria, garantendo che il cluster rimanga entro i limiti operativi richiesti.

La modalità automatica di EKS sfrutta le note etichette Kubernetes per offrire metodi coerenti e standardizzati di identificazione delle caratteristiche dei nodi. Queste etichette, come `topology.kubernetes.io/zone` per le zone di disponibilità e `kubernetes.io/arch` per l'architettura della CPU, seguono le convenzioni stabilite da Kubernetes. Inoltre, EKS-specific

le etichette (precedute da `eks.amazonaws.com/`) estendono questa funzionalità con attributi AWS specifici come tipi di istanze, produttori di CPU, capacità della GPU e specifiche di rete. Questo sistema di etichettatura standardizzato consente una perfetta integrazione con gli strumenti Kubernetes esistenti fornendo al contempo una profonda integrazione dell'infrastruttura. AWS

## Crea un NodePool

Segui questi passaggi per creare un NodePool cluster Amazon EKS:

1. Crea un file YAML denominato `nodepool.yaml` con la configurazione richiesta. NodePool Puoi utilizzare la configurazione di esempio seguente.
2. Applicalo NodePool al tuo cluster:

```
kubectl apply -f nodepool.yaml
```

3. Verifica che sia NodePool stato creato correttamente:

```
kubectl get nodepools
```

4. (Facoltativo) Monitora lo NodePool stato:

```
kubectl describe nodepool default
```

Assicurati che i tuoi NodePool riferimenti NodeClass siano validi e presenti nel tuo cluster. NodeClass Definisce configurazioni AWS specifiche per le tue risorse di calcolo. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

## Esempio NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        billing-team: my-team
    spec:
      nodeClassRef:
```

```
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["4", "8", "16", "32"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["arm64", "amd64"]

  disruption:
    consolidationPolicy: Balanced
    consolidateAfter: 30s

  limits:
    cpu: "1000"
    memory: 1000Gi
```

## Etichette supportate dalla modalità automatica di EKS

La modalità automatica di EKS supporta le seguenti etichette note.

> **Note**
>
> La modalità automatica di EKS utilizza etichette diverse rispetto a Karpenter. Le etichette relative alle istanze gestite da EC2 iniziano con `eks.amazonaws.com`.

| Etichetta | Esempio | Description |
| --- | --- | --- |
| topologia.kubernetes. io/zone | us-east-2a | AWS regione |

| Etichetta | Esempio | Description |
| --- | --- | --- |
| nodo.kubernetes. io/instance-tipo | g4dn.8xlarge | AWS tipo di istanza |
| kubernetes. io/arch | amd64 | Le architetture sono definite dai valori GOARCH sull'istanza |
| falegname. sh/capacity-tipo | spot | I tipi di capacità includonospot,on-demand , ereserved. Se non si specifica un tipo di capacità, Karpenter dà la priorità a, thenreserved, then. spot on-demand |
| eks.amazonaws. com/instance-hypervisor | nitro | Tipi di istanze che usano un hypervisor specifico |
| ex.amazonaws. com/compute-tipo | auto | Identifica i nodi gestiti dalla modalità automatica di EKS |
| eks.amazonaws. com/instance-supportata la crittografia in transito | true | Tipi di istanza che supportano (o non supportan o) la crittografia in transito |
| ex.amazonaws. com/instance-categoria | g | Tipi di istanza della stessa categoria, in genere la stringa che precede il numero di generazione |
| eks.amazonaws. com/instance-generazione | 4 | Numero di generazione del tipo di istanza in una categoria di istanza |
| eks.amazonaws. com/instance-famiglia | g4dn | Tipi di istanza con proprietà simili ma quantità di risorse differenti |
| eks.amazonaws. com/instance-taglia | 8xlarge | Tipi di istanza con quantità di risorse simili ma proprietà differenti |
| ex. amazonaws. com/instance-tazza | 32 | Numero di CPU sull'istanza |
| ex.amazonaws. com/instance-produttore di CPU | aws | Nome del produttore della CPU |

| Etichetta | Esempio | Description |
| --- | --- | --- |
| es. amazonaws. com/instance-memoria | 131072 | Numero di mebibyte di memoria sull'istanza |
| ex. amazonaws. com/instance-ebs-larghezza di banda | 9500 | Numero massimo di megabit di EBS disponibili sull'istanza |
| ex. amazonaws. com/instance-larghezza di banda della rete | 131072 | Numero di megabit di base disponibili sull'istanza |
| ex. amazonaws. com/instance-nome-gpu | t4 | Nome della GPU nell'istanza, se disponibile |
| ex.amazonaws. com/instance-produttore di gpu | nvidia | Nome del produttore della GPU |
| es. amazonaws. com/instance-gpu-count | 1 | Numero di GPU nell'istanza |
| es. amazonaws. com/instance-gpu-memoria | 16384 | Numero di mebibyte di memoria sulla GPU |
| es. amazonaws. com/instance-nome-locale | 900 | Gibibyte di storage nvme locale nell'istanza |
| ex.amazonaws. com/capacity-prenotazione-interrompibile | true | Se la prenotazione della capacità è interrompibile. Presente solo sui nodi con. `karpenter.sh/capacity-type: reserved` |

**Note**

La modalità automatica di EKS supporta solo certe istanze e prevede dei requisiti minimi di dimensione. Per ulteriori informazioni, consulta the section called "Riferimento all'istanza supportata dalla modalità automatica EKS".

### Etichette non supportate dalla modalità automatica di EKS

La modalità automatica di EKS non supporta le seguenti etichette.

- La modalità automatica di EKS supporta soltanto Linux
  - `node.kubernetes.io/windows-build`
  - `kubernetes.io/os`

### Disattivare i pool di nodi integrati

Se crei pool di nodi personalizzati, puoi disattivare i pool di nodi integrati. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

### Cluster senza pool di nodi integrati

Puoi creare un cluster senza pool di nodi integrati. È utile quando l'organizzazione ha creato pool di nodi personalizzati.

> **Note**
>
> Quando si crea un cluster senza pool di nodi integrati, il provisioning non `default` NodeClass viene eseguito automaticamente. Dovrai crearne uno personalizzato NodeClass. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

Panoramica:

1. Crea un cluster EKS con entrambi `nodePools` i `nodeRoleArn` valori vuoti.
   - Esempio eksctl `autoModeConfig`:

     ```
     autoModeConfig:
       enabled: true
       nodePools: []
       # Do not set a nodeRoleARN
     ```

     Per ulteriori informazioni, consulta the section called "CLI di eksctl"
2. Crea una classe di nodi personalizzata con un ruolo del nodo ARN
   - Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi"

3. Crea una voce di accesso per la classe di nodi personalizzata
   - Per ulteriori informazioni, consulta the section called "Creazione di una voce di accesso alla classe di nodi"
4. Crea un pool di nodi personalizzato come descritto sopra.

## Interruzione

È possibile configurare EKS Auto Mode per interrompere i nodi NodePool in diversi modi. Puoi usare `spec.disruption.consolidationPolicy`, `spec.disruption.consolidateAfter` o `spec.template.spec.expireAfter`. È inoltre possibile limitare le interruzioni della modalità EKS Auto tramite gli NodePool». `spec.disruption.budgets` Puoi inoltre controllare le finestre temporali e il numero di nodi simultanei interrotti. Per istruzioni sulla configurazione di questo comportamento, consulta la pagina Disruption nella documentazione di Karpenter.

Il `consolidationPolicy` campo accetta tre valori:

- `WhenEmpty`— Consolida solo i nodi vuoti, in modo che i carichi di lavoro in esecuzione non vengano mai interrotti.
- `WhenEmptyOrUnderutilized`— Si consolida in modo aggressivo in termini di costi, interrompendo i carichi di lavoro ogni volta che i pod di un nodo possono essere reimballati in modo più economico.
- `Balanced`— Assegna un punteggio a ciascuna azione di consolidamento soppesando i costi legati alle interruzioni e i risparmi sui costi. Salta le azioni in cui l'interruzione supera i risparmi.

Per i dettagli su ciascun valore, consulta Disruption nella documentazione di Karpenter.

Puoi configurare l'interruzione per i pool di nodi per:

- Identifica le istanze sottoutilizzate e consolida i carichi di lavoro.
- Crea un budget per le interruzioni dei pool di nodi per limitare le interruzioni dei nodi causate da deviazioni, vuoto e consolidamento.

Per impostazione predefinita, la modalità automatica di EKS:

- Consolida le istanze sottoutilizzate.
- Termina le istanze dopo 336 ore.

- Imposta un budget unico per le interruzioni pari al 10% dei nodi.
- Permette la sostituzione dei nodi a causa della deriva quando viene rilasciata una nuova AMI in Auto Mode, cosa che avviene circa una volta alla settimana.

### Periodo di tolleranza della terminazione

Quando a non `terminationGracePeriod` è definito esplicitamente in una modalità EKS Auto NodePool, il sistema applica automaticamente un periodo di tolleranza di risoluzione predefinito di 24 ore al soggetto associato. NodeClaim Sebbene i clienti di EKS Auto Mode non vedano un `terminationGracePeriod` valore predefinito nelle loro NodePool configurazioni personalizzate, osserveranno questo valore predefinito su. NodeClaim La funzionalità rimane costante indipendentemente dal fatto che il periodo di tolleranza sia impostato esplicitamente su NodePool o predefinito su NodeClaim, garantendo un comportamento prevedibile di terminazione dei nodi in tutto il cluster.

## Pool di nodi a capacità statica in modalità EKS Auto

La modalità automatica di Amazon EKS supporta pool di nodi a capacità statica che mantengono un numero fisso di nodi indipendentemente dalla domanda del pod. I pool di nodi con capacità statica sono utili per carichi di lavoro che richiedono capacità prevedibile, istanze riservate o requisiti di conformità specifici in cui è necessario mantenere un ingombro dell'infrastruttura coerente.

A differenza dei pool di nodi dinamici che si ridimensionano in base alle richieste di pianificazione dei pod, i pool di nodi con capacità statica mantengono il numero di nodi configurati.

### Configurare un pool di nodi con capacità statica

Per creare un pool di nodi con capacità statica, imposta il `replicas` campo nelle tue NodePool specifiche. Il `replicas` campo definisce il numero esatto di nodi che il pool di nodi manterrà. Scopri the section called “Esempi” come configurare`replicas`.

### Considerazioni sulla capacità statica del pool di nodi

I pool di nodi con capacità statica presentano diversi vincoli e comportamenti importanti:

Vincoli di configurazione:

- Impossibile cambiare modalità: una volta impostato `replicas` un pool di nodi, non è possibile rimuoverlo. Il pool di nodi non può passare dalla modalità statica a quella dinamica.

- Limiti di risorse limitati: nella sezione dei limiti è supportato solo il `limits.nodes` campo. I limiti di CPU e memoria non sono applicabili.
- Nessun campo di peso: il `weight` campo non può essere impostato su pool di nodi con capacità statica poiché la selezione dei nodi non si basa sulla priorità.

Comportamento operativo:

- Nessun consolidamento: i nodi nei pool di capacità statici non vengono presi in considerazione per il consolidamento.
- Operazioni di scalabilità: le operazioni di scalabilità evitano i budget relativi alle interruzioni dei nodi, ma rispettano comunque i requisiti. PodDisruptionBudgets
- Sostituzione dei nodi: i nodi vengono comunque sostituiti per eventuali variazioni (come gli aggiornamenti AMI) e per scadenza in base alla configurazione.

## Best practice

Pianificazione della capacità:

- Impostato su un `limits.nodes` valore superiore `replicas` a per consentire il ridimensionamento temporaneo durante le operazioni di sostituzione dei nodi.
- Quando imposti i limiti, considera la capacità massima necessaria durante la deriva dei nodi o gli aggiornamenti AMI.

Selezione dell'istanza:

- Utilizza tipi di istanze specifici quando hai istanze riservate o requisiti hardware specifici.
- Evita requisiti eccessivamente restrittivi che potrebbero limitare la disponibilità delle istanze durante la scalabilità.

Gestione delle interruzioni:

- Configura i budget appropriati per le interruzioni per bilanciare la disponibilità con le operazioni di manutenzione.
- Considerate la tolleranza della vostra applicazione per la sostituzione dei nodi quando impostate le percentuali di budget.

Monitoraggio:

- Monitora regolarmente il `status.nodes` campo per garantire il mantenimento della capacità desiderata.
- Imposta avvisi per quando il numero effettivo di nodi si discosta dalle repliche desiderate.

Distribuzione delle zone:

- Per un'elevata disponibilità, distribuisci la capacità statica su più zone di disponibilità.
- Quando si crea un pool di nodi con capacità statica che si estende su più zone di disponibilità, EKS Auto Mode distribuisce i nodi tra le zone specificate, ma non è garantito che la distribuzione sia uniforme.
- Per una distribuzione prevedibile e uniforme tra le zone di disponibilità, crea pool di nodi con capacità statica separati, ciascuno collegato a una zona di disponibilità specifica utilizzando il requisito. `topology.kubernetes.io/zone`
- Se hai bisogno di 12 nodi distribuiti uniformemente su tre zone, crea tre pool di nodi con 4 repliche ciascuno, anziché un pool di nodi con 12 repliche su tre zone.

## Scalabilità di un pool di nodi con capacità statica

È possibile modificare il numero di repliche in un pool di nodi con capacità statica utilizzando il `kubectl scale` comando:

```
# Scale down to 5 nodes
kubectl scale nodepool static-nodepool --replicas=5
```

In caso di ridimensionamento, EKS Auto Mode terminerà i nodi in modo corretto, rispettando PodDisruptionBudgets e consentendo la riprogrammazione dei pod in esecuzione sui nodi rimanenti.

## Monitora i pool di nodi con capacità statica

Utilizzate i seguenti comandi per monitorare i pool di nodi con capacità statica:

```
# View node pool status
kubectl get nodepool static-nodepool

# Get detailed information including current node count
kubectl describe nodepool static-nodepool
```

```
# Check the current number of nodes
kubectl get nodepool static-nodepool -o jsonpath='{.status.nodes}'
```

Il `status.nodes` campo mostra il numero attuale di nodi gestiti dal pool di nodi, che dovrebbe corrispondere al `replicas` numero desiderato in condizioni normali.

## Risoluzione dei problemi

I nodi non raggiungono le repliche desiderate:

- Controlla se il `limits.nodes` valore è sufficiente
- Verifica che i requisiti non limitino eccessivamente la selezione delle istanze
- Controlla le quote AWS di servizio per i tipi di istanze e le regioni che stai utilizzando

La sostituzione dei nodi richiede troppo tempo:

- Modifica i budget per le interruzioni per consentire più sostituzioni simultanee
- Controlla se PodDisruptionBudgets stanno impedendo la chiusura del nodo

Chiusura imprevista del nodo:

- Rivedi le impostazioni `expireAfter` e `terminationGracePeriod`
- Verifica la presenza di interruzioni manuali dei nodi o di eventi AWS di manutenzione

## Esempi

Pool di nodi con capacità statica di base

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: basic-static
spec:
  replicas: 5

  template:
    spec:
      nodeClassRef:
```

```
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["m"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a"]

  limits:
    nodes: 8  # Allow scaling up to 8 during operations
```

### Capacità statica con tipi di istanze specifici

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: reserved-instances
spec:
  replicas: 20

  template:
    metadata:
      labels:
        instance-type: reserved
        cost-center: production
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "node.kubernetes.io/instance-type"
          operator: In
          values: ["m5.2xlarge"]  # Specific instance type
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "topology.kubernetes.io/zone"
```

```
          operator: In
          values: ["us-west-2a", "us-west-2b", "us-west-2c"]

  limits:
    nodes: 25

  disruption:
    # Conservative disruption for production workloads
    budgets:
      - nodes: 10%
```

### Pool di nodi con capacità statica multizona

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: multi-zone-static
spec:
  replicas: 12  # Will be distributed across specified zones

  template:
    metadata:
      labels:
        availability: high
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["8", "16"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b", "us-west-2c"]
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
```

```
  limits:
    nodes: 15

  disruption:
    budgets:
      - nodes: 25%
```

Capacità statica con riserva di capacità

L'esempio seguente mostra come utilizzare un pool di nodi di capacità statica con una prenotazione di capacità EC2. Per ulteriori informazioni sull'utilizzo di EC2 Capacity Reservations con EKS Auto Mode, consulta. the section called "Controllare le prenotazioni della capacità"

`NodeClass`definizione del `capacityReservationSelectorTerms`

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: capacity-reservation-nodeclass
spec:
  role: AmazonEKSNodeRole
  securityGroupSelectorTerms:
  - id: sg-0123456789abcdef0
  subnetSelectorTerms:
  - id: subnet-0123456789abcdef0
  capacityReservationSelectorTerms:
  - id: cr-0123456789abcdef0
```

`NodePool`facendo riferimento a quanto sopra `NodeClass` e utilizzando`karpenter.sh/capacity-type: reserved`.

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: static-capacity-reservation-nodepool
spec:
  replicas: 5
  limits:
    nodes: 8  # Allow scaling up to 8 during operations
  template:
    metadata: {}
    spec:
```

```
    nodeClassRef:
      group: eks.amazonaws.com
      kind: NodeClass
      name: capacity-reservation-nodeclass
    requirements:
    - key: karpenter.sh/capacity-type
      operator: In
      values: ['reserved']
```

## Creare e IngressClass configurare un Application Load Balancer

La modalità automatica di EKS automatizza le attività di routine per il bilanciamento del carico, inclusa l'esposizione delle app del cluster a Internet.

AWS suggerisce di utilizzare Application Load Balancer (ALB) per gestire il traffico HTTP e HTTPS. Gli Application Load Balancer possono instradare le richieste in base al contenuto della richiesta. Per ulteriori informazioni sugli Application Load Balancer, consulta What is Elastic Load Balancing?

La modalità automatica di EKS crea e configura Application Load Balancer (ALB). Ad esempio, EKS Auto Mode crea un load balancer quando si crea un oggetto `Ingress` Kubernetes e lo configura per indirizzare il traffico verso il carico di lavoro del cluster.

Panoramica

1. Crea un carico di lavoro da esporre a Internet.
2. Crea una `IngressClassParams` risorsa, specificando valori di configurazione AWS specifici come il certificato da utilizzare e le sottoreti VPC. SSL/TLS
3. Creare una risorsa `IngressClass`, specificando che modalità automatica di EKS sarà il controller della risorsa.
4. Crea una `Ingress` risorsa che associ un percorso e una porta HTTP a un carico di lavoro del cluster.

La modalità automatica di EKS creerà un Application Load Balancer che punta al carico di lavoro specificato nella risorsa `Ingress`, utilizzando la configurazione del bilanciatore del carico specificata nella risorsa `IngressClassParams`.

### Prerequisiti

- La modalità automatica di EKS su un cluster Amazon EKS

- Kubectl configurato per connettersi al cluster
  - Puoi usare `kubectl apply -f <filename>` per applicare i file di configurazione di esempio YAML riportati di seguito al cluster.

> **Note**
>
> La modalità automatica di EKS richiede i tag di sottorete per identificare le sottoreti pubbliche e private.
> Se hai creato il cluster con `eksctl`, disponi già di questi tag.
> Informazioni su come the section called "Assegnazione di tag alle sottoreti".

## Fase 1: Creare un carico di lavoro

Per iniziare, crea un carico di lavoro da esporre a Internet. Può trattarsi di qualsiasi risorsa di Kubernetes che serve il traffico HTTP, ad esempio un'implementazione o un servizio.

Questo esempio utilizza un semplice servizio HTTP chiamato `service-2048` che ascolta sulla porta `80`. Creare questo servizio e la sua implementazione applicando il seguente manifesto, `2048-deployment-service.yaml`:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 2
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
```

```
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: app-2048
```

Applica la configurazione al cluster:

```
kubectl apply -f 2048-deployment-service.yaml
```

Le risorse sopra elencate verranno create nel namespace predefinito. Puoi verificarlo eseguendo il comando seguente:

```
kubectl get all -n default
```

## Fase 2: Creazione IngressClassParams

Crea un `IngressClassParams` oggetto per specificare opzioni di configurazione AWS specifiche per Application Load Balancer. In questo esempio, creiamo una risorsa `IngressClassParams` denominata `alb` (che verrà utilizzata nella fase successiva) che specifica lo schema del bilanciatore del carico come `internet-facing` in un file chiamato `alb-ingressclassparams.yaml`.

```
apiVersion: eks.amazonaws.com/v1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing
```

Applica la configurazione al cluster:

```
kubectl apply -f alb-ingressclassparams.yaml
```

### Fase 3: Creazione IngressClass

Crea un `IngressClass` modulo che faccia riferimento ai valori di configurazione AWS specifici impostati nella `IngressClassParams` risorsa in un file denominato`alb-ingressclass.yaml`. Annotare il nome della `IngressClass`. In questo esempio, sia `IngressClass` che `IngressClassParams` sono denominati `alb`.

Utilizzare l'annotazione `is-default-class` per controllare se le risorse `Ingress` devono utilizzare questa classe per impostazione predefinita.

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    # Use this annotation to set an IngressClass as Default
    # If an Ingress doesn't specify a class, it will use the Default
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  # Configures the IngressClass to use EKS Auto Mode
  controller: eks.amazonaws.com/alb
  parameters:
    apiGroup: eks.amazonaws.com
    kind: IngressClassParams
    # Use the name of the IngressClassParams set in the previous step
    name: alb
```

Per ulteriori informazioni sulle opzioni di configurazione, consultare the section called "IngressClassParams Riferimento".

Applica la configurazione al cluster:

```
kubectl apply -f alb-ingressclass.yaml
```

### Fase 4: Creare Ingress

Creare una risorsa `Ingress` in un file denominato `alb-ingress.yaml`. Lo scopo di questa risorsa è associare percorsi e porte sull'Application Load Balancer ai carichi di lavoro del cluster. Per questo

esempio, creiamo una risorsa `Ingress` denominata `2048-ingress` che instrada il traffico verso un servizio denominato `service-2048` sulla porta 80.

Per ulteriori informazioni sulla configurazione di questa risorsa, consultare Ingress nella documentazione su Kubernetes.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: 2048-ingress
spec:
  # this matches the name of IngressClass.
  # this can be omitted if you have a default ingressClass in cluster: the one with
 ingressclass.kubernetes.io/is-default-class: "true"  annotation
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Applica la configurazione al cluster:

```
kubectl apply -f alb-ingress.yaml
```

## Fase 5: Controllare lo stato

Utilizza `kubectl` per trovare lo stato di `Ingress`. Prima che il bilanciatore del carico sia disponibile, potrebbero essere necessari alcuni minuti.

Utilizza il nome della risorsa `Ingress` impostata nella fase precedente. Ad esempio:

```
kubectl get ingress 2048-ingress
```

Una volta che la risorsa è pronta, recuperare il nome di dominio del bilanciatore del carico.

```
kubectl get ingress 2048-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

Per visualizzare il servizio in un browser Web, esaminate la porta e il percorso specificati nella `Ingress` risorsa.

### Fase 6: Eliminazione

Per eliminare il bilanciatore del carico, utilizza il seguente comando:

```
kubectl delete ingress 2048-ingress
kubectl delete ingressclass alb
kubectl delete ingressclassparams alb
```

EKS Auto Mode eliminerà automaticamente il load balancer associato nel tuo AWS account.

### IngressClassParams Riferimento

La tabella seguente è un riferimento rapido per le opzioni di configurazione più comunemente utilizzate.

| Campo | Description | Valore di esempio |
|---|---|---|
| `scheme` | Definisce se l'ALB è interno o esposto a Internet | `internet-facing` |
| `namespaceSelector` | Limita i namespace che possono utilizzarlo IngressCl ass | `environment: prod` |
| `group.name` | Raggruppa più ingressi per condividere un singolo ALB | `retail-apps` |
| `ipAddressType` | Imposta il tipo di indirizzo IP per ALB | `dualstack` |
| `subnets.ids` | Elenco di ID di sottorete per l'implementazione di ALB | `subnet-xxxx, subnet-yyyy` |

| Campo | Description | Valore di esempio |
|---|---|---|
| `subnets.tags` | Filtri di tag per selezionare le sottoreti per ALB | `Environment: prod` |
| `certificateARNs` | ARN di certificati SSL da utilizzare | `arn:aws: acm:region:account:certificate/id` |
| `tags` | Tag personalizzati per le risorse AWS | `Environment: prod, Team: platform` |
| `loadBalancerAttributes` | Attributi specifici del bilanciatore del carico | `idle_timeout.timeout_seconds: 60` |

## Considerazioni

- Non è possibile utilizzare Annotations su e IngressClass per configurare i sistemi di bilanciamento del carico con la modalità automatica EKS. IngressClass la configurazione deve essere effettuata tramite. IngressClassParams Tuttavia, è possibile utilizzare le annotazioni sulle singole risorse Ingress per configurare il comportamento del sistema di bilanciamento del carico (ad esempio `alb.ingress.kubernetes.io/security-group-prefix-lists` o). `alb.ingress.kubernetes.io/conditions.*`
- Non è possibile eseguire l'impostazione ListenerAttribute con EKS Auto Mode.
- È necessario aggiornare il ruolo IAM del cluster per abilitare la propagazione dei tag da Kubernetes alle risorse Load AWS Balancer. Per ulteriori informazioni, consulta the section called "Personalizzato AWS tag per le risorse EKS Auto".
- Per informazioni sull'associazione delle risorse a EKS Auto Mode o al Load Balancer Controller AWS autogestito, vedere. the section called "Materiale di riferimento per le migrazioni"
- Per informazioni sulla risoluzione dei problemi con i bilanciatori del carico, consulta the section called "Risoluzione dei problemi".
- Per ulteriori considerazioni sull'utilizzo della funzionalità di bilanciamento del carico della modalità automatica di EKS, consulta the section called "Sistema di bilanciamento del carico".

Le tabelle seguenti forniscono un confronto dettagliato delle modifiche IngressClassParams, delle annotazioni di Ingress e delle TargetGroupBinding configurazioni per EKS Auto Mode. Queste tabelle evidenziano le principali differenze tra la capacità di bilanciamento del carico della modalità automatica di EKS e il controller del bilanciatore del carico open source, comprese le modifiche alla versione dell'API, le funzionalità obsolete e i nomi dei parametri aggiornati.

IngressClassParams

| Precedente | Novità | Description |
|---|---|---|
| `elbv2.k8s.aws/v1beta1` | `eks.amazonaws.com/v1` | Modifica alla versione dell'API |
| `spec.certificateArn` | `spec.certificateARNs` | Supporto per più ARN dei certificati |
| `spec.subnets.tags` | `spec.subnets.matchTags` | Schema di corrispondenza delle sottoreti modificato |
| `spec.listeners.listenerAttributes` | Non supportata | Non ancora supportato dalla modalità automatica di EKS |

Annotazioni di ingresso

| Precedente | Novità | Description |
|---|---|---|
| `kubernetes.io/ingress.class` | Non supportata | Utilizzare `spec.ingressClassName` su oggetti di ingresso |
| `alb.ingress.kubernetes.io/group.name` | Non supportata | Specifica i gruppi solo in IngressClass |
| `alb.ingress.kubernetes.io/waf-acl-id` | Non supportata | Usare invece WAF v2 |
| `alb.ingress.kubernetes.io/web-acl-id` | Non supportata | Usare invece WAF v2 |

| Precedente | Novità | Description |
| --- | --- | --- |
| `alb.ingress.kubernetes.io/wafv2-acl-name` | Non supportata | Prossimamente |
| `alb.ingress.kubernetes.io/dry-run-plan` | Non supportata | Dry-run il piano non è attualmente supportato |
| `alb.ingress.kubernetes.io/create-acm-cert` | Non supportata | La creazione di certificati ACM non è attualmente supportata |
| `alb.ingress.kubernetes.io/acm-pca-arn` | Non supportata | ACM PCA ARN non è attualmente supportato |
| `alb.ingress.kubernetes.io/auth-type: oidc` | Non supportata | Il tipo di autenticazione OIDC non è attualmente supportato |

TargetGroupBinding

| Precedente | Novità | Description |
| --- | --- | --- |
| `elbv2.k8s.aws/v1beta1` | `eks.amazonaws.com/v1` | Modifica alla versione dell'API |
| `spec.targetType` facoltativo | `spec.targetType` obbligatorio | Specifica esplicita del tipo di destinazione |
| `spec.networking.ingress.from` | Non supportata | Non supporta più NLB senza gruppi di sicurezza |

Per utilizzare la TargetGroupBinding funzionalità personalizzata, è necessario contrassegnare il gruppo target con il tag eks:eks-cluster-name con il nome del cluster per concedere al controller le

autorizzazioni IAM necessarie. Tieni presente che il controller eliminerà il gruppo di destinazione quando la risorsa o il cluster vengono eliminati TargetGroupBinding .

## Usa Annotazioni del servizio per configurare Network Load Balancer

Scopri come configurare Network Load Balancer (NLB) in Amazon EKS utilizzando le annotazioni del servizio di Kubernetes. Questo argomento spiega le annotazioni supportate da EKS Auto Mode per personalizzare il comportamento NLB, tra cui l'accessibilità a Internet, i controlli di integrità, la SSL/TLS terminazione e le modalità di targeting IP.

Quando si crea un servizio Kubernetes di tipo EKS Auto Mode, EKS esegue automaticamente il provisioning e configura un Network Load Balancer `LoadBalancer` in base alle annotazioni specificate. AWS Questo approccio dichiarativo ti consente di gestire le configurazioni del bilanciatore del carico direttamente tramite i manifesti di Kubernetes, mantenendo l'infrastruttura come pratiche di codice.

EKS Auto Mode gestisce il provisioning di Network Load Balancer per impostazione predefinita per tutti i servizi di tipo LoadBalancer : non è richiesta l'installazione o la configurazione di controller aggiuntivi. La specifica `loadBalancerClass: eks.amazonaws.com/nlb` viene impostata automaticamente come impostazione predefinita del cluster, semplificando il processo di implementazione e mantenendo la compatibilità con i carichi di lavoro di Kubernetes esistenti.

> **Note**
>
> La modalità automatica di EKS richiede i tag di sottorete per identificare le sottoreti pubbliche e private.
> Se hai creato il cluster con `eksctl`, disponi già di questi tag.
> Informazioni su come the section called "Assegnazione di tag alle sottoreti".

### Servizio di esempio

Per ulteriori informazioni sulla risorsa `Service` di Kubernetes, consulta Kubernetes Documentation.

Consulta la risorsa di esempio `Service` riportata di seguito:

```
apiVersion: v1
kind: Service
metadata:
  name: echoserver
```

```
  annotations:
    # Specify the load balancer scheme as internet-facing to create a public-facing
 Network Load Balancer (NLB)
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  selector:
    app: echoserver
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: LoadBalancer
  # Specify the new load balancer class for NLB as part of EKS Auto Mode feature
  # For clusters with Auto Mode enabled, this field can be omitted as it's the default
  loadBalancerClass: eks.amazonaws.com/nlb
```

## Annotazioni di uso comune

La tabella seguente elenca le annotazioni supportate per Amazon EKS Auto Mode. Amazon EKS Auto Mode non supporta tutte le annotazioni di servizio disponibili nel Load AWS Balancer Controller.

> **Tip**
>
> Tutte le seguenti annotazioni devono avere il prefisso `service.beta.kubernetes.io/`

| Campo | Description | Esempio |
|---|---|---|
| `aws-load-balancer-type` | Specifica il tipo di bilanciatore del carico. Utilizza `external` per nuove implementazioni. | `external` |
| `aws-load-balancer-nlb-target-type` | Specifica se instradare il traffico verso le istanze del nodo o direttamente verso gli IP dei pod. Utilizza `instance` per implementazioni standard o `ip` per l'instradamento diretto ai pod. | `instance` |

| Campo | Description | Esempio |
| --- | --- | --- |
| `aws-load-balancer-scheme` | Controlla se il bilanciatore del carico è interno o esposto a Internet. | `internet-facing` |
| `aws-load-balancer-healthcheck-protocol` | Protocollo di controllo dell'integrità per il gruppo di destinazione. Le opzioni più comuni sono `TCP` (impostazione predefinita) o `HTTP`. | `HTTP` |
| `aws-load-balancer-healthcheck-path` | Il percorso HTTP per i controlli sullo stato di salute quando si utilizza HTTP/HTTPS il protocollo. | `/healthz` |
| `aws-load-balancer-healthcheck-port` | Porta utilizzata per i controlli dell'integrità. Può essere un numero di porta specifico o `traffic-port` . | `traffic-port` |
| `aws-load-balancer-subnets` | Specifica in quali sottoreti creare il bilanciatore del carico. Può utilizzare ID o nomi di sottorete. | `subnet-xxxx, subnet-yyyy` |
| `aws-load-balancer-ssl-cert` | ARN del certificato SSL di Certificate AWS Manager per. HTTPS/TLS | `arn:aws: acm:region:account:certificate/cert-id` |
| `aws-load-balancer-ssl-ports` | Specifica quali porte devono essere utilizzate. SSL/TLS | `443, 8443` |
| `load-balancer-source-ranges` | Intervalli CIDR autorizzati ad accedere al bilanciatore del carico. | `10.0.0.0/24, 192.168.1.0/24` |

| Campo | Description | Esempio |
| --- | --- | --- |
| `aws-load-balancer-additional-resource-tags` | AWS Tag aggiuntivi da applicare al load balancer e alle risorse correlate. | `Environment=prod,Team=platform` |
| `aws-load-balancer-ip-address-type` | Specifica se il bilanciatore del carico utilizza IPv4 o doppio stack (IPv4 + IPv6). | `ipv4` o `dualstack` |
| `aws-load-balancer-listener-attributes.${Protocol}-${Port}` | Specifica gli attributi del listener da applicare al listener. | `tcp.idle_timeout.seconds=350` |

## Considerazioni

- È necessario aggiornare il ruolo IAM del cluster per abilitare la propagazione dei tag da Kubernetes alle risorse di Load Balancer. AWS Per ulteriori informazioni, consulta the section called "Personalizzato AWS tag per le risorse EKS Auto".
- Per informazioni sull'associazione delle risorse a EKS Auto Mode o al Load Balancer Controller AWS autogestito, vedere. the section called "Materiale di riferimento per le migrazioni"
- Per informazioni sulla risoluzione dei problemi con i bilanciatori del carico, consulta the section called "Risoluzione dei problemi".
- Per ulteriori considerazioni sull'utilizzo della funzionalità di bilanciamento del carico della modalità automatica di EKS, consulta the section called "Sistema di bilanciamento del carico".

Durante la migrazione alla modalità automatica di EKS per il bilanciamento del carico, sono necessarie diverse modifiche nelle annotazioni dei servizi e nelle configurazioni delle risorse. Le tabelle seguenti descrivono le differenze principali tra le implementazioni precedenti e quelle nuove, includendo le opzioni non supportate e le alternative consigliate.

Annotazioni dei servizi

| Precedente | Novità | Description |
| --- | --- | --- |
| `service.beta.kubernetes.io/load-balancer-source-ranges` | Non supportata | Usa `spec.loadBalancerSourceRanges` sul servizio |
| `service.beta.kubernetes.io/aws-load-balancer-type` | Non supportata | Usa `spec.loadBalancerClass` sul servizio |
| `service.beta.kubernetes.io/aws-load-balancer-internal` | Non supportata | Utilizzare `service.beta.kubernetes.io/aws-load-balancer-scheme` |
| `service.beta.kubernetes.io/aws-load-balancer-proxy-protocol` | Non supportata | Usa invece `service.beta.kubernetes.io/aws-load-balancer-target-group-attributes` |
| Vari attributi del bilanciatore del carico | Non supportata | Utilizzare `service.beta.kubernetes.io/aws-load-balancer-attributes` |
| `service.beta.kubernetes.io/aws-load-balancer-access-log-enabled` | Non supportata | Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes` |
| `service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name` | Non supportata | Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes` |

| Precedente | Novità | Description |
| --- | --- | --- |
| `service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-prefix` | Non supportata | Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes` |
| `service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled` | Non supportata | Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes` |

Per eseguire la migrazione dalle annotazioni obsolete degli attributi del bilanciatore del carico, consolida queste impostazioni nell'annotazione `service.beta.kubernetes.io/aws-load-balancer-attributes`. Questa annotazione accetta un elenco di coppie chiave-valore separato da virgole per vari attributi del bilanciatore del carico. Ad esempio, per specificare la registrazione degli accessi e il bilanciamento del carico tra zone, utilizzate il seguente formato:

```
service.beta.kubernetes.io/aws-load-balancer-attributes:
 access_logs.s3.enabled=true,access_logs.s3.bucket=my-bucket,access_logs.s3.prefix=my-prefix,load_balancing.cross_zone.enabled=true
```

Questo formato consolidato offre un modo più coerente e flessibile per configurare gli attributi del bilanciatore del carico riducendo al contempo il numero di singole annotazioni necessarie. Rivedi le configurazioni del servizio esistenti e aggiornale per utilizzare questo formato consolidato.

TargetGroupBinding

| Precedente | Novità | Description |
| --- | --- | --- |
| `elbv2.k8s.aws/v1beta1` | `eks.amazonaws.com/v1` | Modifica alla versione dell'API |
| `spec.targetType` facoltativo | `spec.targetType` obbligatorio | Specifica esplicita del tipo di destinazione |

| Precedente | Novità | Description |
| --- | --- | --- |
| `spec.networking.ingress.from` | Non supportata | Non supporta più NLB senza gruppi di sicurezza |

Nota: per utilizzare la TargetGroupBinding funzionalità personalizzata, devi contrassegnare il gruppo di destinazione con il tag con il `eks:eks-cluster-name` nome del cluster per concedere al controller le autorizzazioni IAM necessarie. Tieni presente che il controller eliminerà il gruppo di destinazione quando la TargetGroupBinding risorsa o il cluster vengono eliminati.

## Configurazione di gruppi target multi-cluster

Per impostazione predefinita, il controller di bilanciamento del carico Amazon EKS Auto Mode assume il pieno controllo sui gruppi target che configura. Quando un gruppo target viene registrato presso il controller, annulla la registrazione di tutti i target che non sono attualmente presenti nel cluster. Ciò garantisce che il gruppo target rifletta l'esatto insieme di obiettivi gestiti da quel cluster.

L'attivazione del  MultiCluster  supporto modifica questo comportamento. Un gruppo target con MultiCluster supporto abilitato può essere associato a più cluster Kubernetes o supportare target arbitrari provenienti da altre fonti. Ogni cluster registra e annulla la registrazione solo delle destinazioni di sua proprietà, invece di annullare la registrazione delle destinazioni che appartengono ad altri cluster o fonti.

### Come funziona

Quando MultiCluster il supporto è abilitato, ogni cluster tiene traccia del set di obiettivi che gestisce e riconcilia solo quelli del gruppo target condiviso. Amazon EKS Auto Mode archivia gli obiettivi tracciati di ogni cluster in un `ConfigMap` nome Kubernetes`eks-lbc-targets-<targetgroupbinding-name>`, nello stesso spazio dei nomi delle risorse di bilanciamento del carico.

> **Note**
>
> Il Load AWS Balancer Controller upstream memorizza questo stato con il prefisso. `ConfigMap` `aws-lbc-targets-` In Amazon EKS Auto Mode, il prefisso è `eks-lbc-targets-` invece, quindi quando segui le indicazioni upstream, controlla il. `eks-lbc-targets-` `ConfigMap`

Il load balancer distribuisce il traffico in modo equo su tutte le destinazioni registrate. Il controller non supporta la distribuzione ponderata tra i cluster. MultiCluster il supporto funziona con entrambi i tipi `ip` di `instance` destinazione.

## Abilita MultiCluster il supporto

Abilita l' MultiCluster assistenza utilizzando uno dei seguenti metodi, a seconda del tipo di risorsa:

- TargetGroupBinding— impostato`spec.multiClusterTargetGroup: true`.
- Ingress (ALB): aggiunge l'annotazione. `alb.ingress.kubernetes.io/multi-cluster-target-group: "true"`
- Service (NLB): aggiungi l'annotazione. `service.beta.kubernetes.io/aws-load-balancer-multi-cluster-target-group: "true"`

## Considerazioni per la modalità automatica di Amazon EKS

La guida upstream MultiCluster Target Groups sul GitHub sito web di AWS Load Balancer Controller illustra la condivisione di un gruppo target tra due cluster. Per una discussione più ampia sulle architetture multi-cluster, consulta Creazione di applicazioni multi-cluster resilienti con Amazon EKS sul blog. AWS Lo stesso flusso funziona su Amazon EKS Auto Mode con le seguenti differenze:

- La risorsa `TargetGroupBinding` personalizzata è nel gruppo `eks.amazonaws.com/v1` API, none`elbv2.k8s.aws/v1beta1`.
- Il `IngressClass` controller è`eks.amazonaws.com/alb`.
- Non impostare l'`alb.ingress.kubernetes.io/tags`annotazione. La policy `AmazonEKSLoadBalancingPolicy` gestita si limita `aws:TagKeys` a una lista consentita, pertanto i tag personalizzati causano un errore di accesso negato e il controller non può riconciliarli. Ciò può causare il blocco del finalizzatore Ingress durante l'eliminazione.
- Tutti i cluster partecipanti devono trovarsi nello stesso VPC, poiché un gruppo target è associato a un singolo VPC.

## Esempio

Per una procedura dettagliata che condivide un singolo gruppo target tra due cluster, uno che possiede il load balancer tramite un Ingress e l'altro che si unisce tramite a, consulta. `TargetGroupBinding` the section called “Condividi un gruppo target”

# Condividi un gruppo target su due cluster

L'esempio seguente condivide un gruppo target tra due cluster. Il primo cluster possiede l'Application Load Balancer tramite un Ingress, mentre il secondo cluster si unisce al gruppo target condiviso tramite uno creato manualmente. `TargetGroupBinding` Prima di iniziare, rivedi the section called "Multi-cluster gruppi target" per capire come funziona MultiCluster il supporto e come abilitarlo.

## Cluster 1: gestisci il load balancer tramite Ingress

Distribuisci la tua applicazione (ad esempio, una distribuzione con un `NodePort` servizio), quindi crea un `IngressClass` and an `Ingress` contenente l'annotazione multi-cluster.

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb-mc
spec:
  controller: eks.amazonaws.com/alb
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: echoserver
  namespace: echoserver
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: instance
    alb.ingress.kubernetes.io/multi-cluster-target-group: "true"
spec:
  ingressClassName: alb-mc
  rules:
    - http:
        paths:
          - path: /
            pathType: Exact
            backend:
              service:
                name: echoserver
                port:
                  number: 80
```

Il controller esegue il provisioning del load balancer e genera un with. `TargetGroupBinding` `spec.multiClusterTargetGroup: true` Nota i seguenti valori tratti dai campi Generated `TargetGroupBinding` and Service: il secondo cluster li richiede tutti e tre:

- Tipo `targetGroupARN`.
- Il servizio`nodePort`.
- L'ID del gruppo di sicurezza del load balancer di`spec.networking.ingress[].from[].securityGroup.groupID`.

## Cluster 2: unisciti al gruppo target condiviso tramite un TargetGroupBinding

Distribuisci la stessa applicazione, fissando il servizio al cluster 1 `nodePort` dal cluster 1, quindi crea un ARN `TargetGroupBinding` che faccia riferimento all'ARN del gruppo di destinazione e all'ID del gruppo di sicurezza del cluster 1.

```
apiVersion: eks.amazonaws.com/v1
kind: TargetGroupBinding
metadata:
  name: mc-echoserver
  namespace: echoserver
spec:
  multiClusterTargetGroup: true
  serviceRef:
    name: echoserver
    port: 80
  targetGroupARN: <cluster-1-target-group-arn>
  targetType: instance
  networking:
    ingress:
      - from:
          - securityGroup:
              groupID: <cluster-1-alb-security-group-id>
        ports:
          - port: <cluster-2-node-port>
            protocol: TCP
```

Con MultiCluster il supporto abilitato su entrambi i binding, ogni cluster registra e annulla la registrazione solo i propri obiettivi. I nodi di entrambi i cluster rimangono integri nell'unico gruppo target condiviso e il load balancer distribuisce il traffico su tutti.

# Concedere un RBAC Kubernetes aggiuntivo ai controller gestiti EKS Auto Mode

Amazon EKS Auto Mode gestisce i seguenti controller per tuo conto: bilanciamento del carico, rete, elaborazione, storage a blocchi e informazioni sui cluster. Questi controller funzionano sul piano di controllo EKS. Si autenticano nel cluster come ruolo `AWSServiceRoleForAmazonEKS` collegato al servizio e le loro autorizzazioni all'interno del cluster provengono da politiche di accesso AWS gestite allegate a tale voce di accesso.

Puoi estendere tu stesso il Kubernetes RBAC di un controller gestito. Ciò è utile quando la policy gestita non concede ancora l'autorizzazione richiesta, ad esempio su una specifica. `get Secret`

Nei cluster Amazon EKS Auto Mode, la voce di accesso creata automaticamente per dichiara `AWSServiceRoleForAmazonEKS` anche il gruppo Kubernetes. `eks:managed` Kubernetes RBAC è additivo. A `RoleBinding` (o`ClusterRoleBinding`) il cui `subjects:` nome il `eks:managed` gruppo concede ai controller gestiti autorizzazioni aggiuntive oltre alle politiche di accesso gestito.

## Come funziona

- I controller gestiti EKS Auto Mode si connettono al cluster come ruolo collegato al servizio. `AWSServiceRoleForAmazonEKS`
- La voce di accesso creata automaticamente per quel ruolo dichiara. `kubernetesGroups: ["eks:managed"]`
- Crei un file `Role` (con namespace) o `ClusterRole` (a livello di cluster) che descrive l'autorizzazione aggiuntiva e un/il cui nome è il gruppo. `RoleBinding` `ClusterRoleBinding` `subjects: eks:managed`
- Kubernetes RBAC combina l'associazione con le politiche di accesso gestito, in modo che i controller gestiti ricevano entrambi i set di autorizzazioni.

Per confermare la presenza del gruppo, esegui:

```
CLUSTER=<your-cluster>
REGION=<your-region>
ACCT=$(aws sts get-caller-identity --query Account --output text)

aws eks describe-access-entry \
  --cluster-name "$CLUSTER" --region "$REGION" \
```

```
  --principal-arn "arn:aws:iam::${ACCT}:role/aws-service-role/eks.amazonaws.com/
AWSServiceRoleForAmazonEKS" \
  --query 'accessEntry.kubernetesGroups'
```

Output previsto:

```
[
    "eks:managed"
]
```

## Ambito ristretto della sovvenzione

Il `eks:managed` gruppo si applica a tutti i controller gestiti EKS Auto Mode. Ambita la concessione aggiuntiva nel modo più ristretto consentito dal tuo caso d'uso, in modo da esporre solo esattamente ciò di cui ha bisogno un controller gestito:

- Preferisci `Role` + `RoleBinding` (namespace) a + (a livello di cluster). `ClusterRole` `ClusterRoleBinding`
- Limita `resourcesresourceNames`, e `verbs` al set più piccolo che sblocca il tuo caso d'uso. Ad esempio, concedi `get` su un singolo oggetto denominato `Secret` anziché `list, watch` su tutti `Secret` gli oggetti.
- Elimina gli `Role` e `RoleBinding` quando non sono più necessari.

## Considerazioni per la modalità automatica di Amazon EKS

- Questo meccanismo estende solo il Kubernetes RBAC dei controller gestiti EKS Auto Mode. Non modifica le autorizzazioni AWS IAM concesse dalle politiche di accesso gestite.
- Il `eks:managed` gruppo è presente sui cluster EKS Auto Mode il cui ruolo collegato ai servizi è. `AWSServiceRoleForAmazonEKS`
- Un'associazione a `eks:managed` si applica a tutti i controller gestiti EKS Auto Mode. Per mantenere piccolo il pratico raggio di esplosione, usate oggetti con `resourceNames` `Role` namespace. Questo espone solo le risorse che nomini e solo attraverso i verbi che concedi.
- Si tratta di un modello ponte destinato ai casi in cui una politica di accesso gestito non copre ancora l'autorizzazione necessaria. Se una policy AWS gestita copre già l'autorizzazione, utilizza invece la policy gestita.

### Esempio

Per una procedura dettagliata che concede al controller di bilanciamento del carico EKS Auto Mode un OIDC `Secret` denominato (sbloccando l'annotazione `get` su un ALB), consulta. `alb.ingress.kubernetes.io/auth-type: oidc` `Ingress` the section called “Concedi l'accesso segreto a LBC”

## Concedi al controller di bilanciamento del carico EKS Auto Mode l'accesso a un segreto specifico

In questo tutorial, concedi al controller di bilanciamento del carico Amazon EKS Auto Mode l'accesso in lettura a un Kubernetes specifico. `Secret` Questo sblocca l'autenticazione OIDC su un ALB. `Ingress` Si utilizza un namespace `Role` con ambito e destinato al gruppo. `RoleBinding` `eks:managed`

Il completamento di questo tutorial richiede circa 15 minuti. Gli oggetti Kubernetes RBAC creati non comportano costi. AWS

Il controller di bilanciamento del carico Amazon EKS Auto Mode supporta l'autenticazione OIDC sugli oggetti ALB. `Ingress` Utilizza l'`alb.ingress.kubernetes.io/auth-type: oidc`annotazione per abilitare questa autenticazione. Per risolvere la configurazione OIDC, il controller legge un `Secret` Kubernetes denominato nell'annotazione. `alb.ingress.kubernetes.io/auth-idp-oidc` La sua politica di accesso gestito attualmente non consente la concessione `Secret` degli oggetti, quindi `get` il riconciliatore fallisce con un messaggio simile a:

```
Failed build model due to ingress: <ns>/<name>:
  secrets "<secret>" is forbidden:
  User "arn:aws:sts::<acct>:assumed-role/AWSServiceRoleForAmazonEKS/<session>"
  cannot get resource "secrets" in API group "" in the namespace "<ns>"
```

Utilizzando il meccanismo descritto inthe section called “Concedi RBAC ai controller gestiti”, è possibile concedere al controller gestito esattamente l'`getSecret`OIDC di cui ha bisogno.

### Prerequisiti

Prima di iniziare, assicurati di avere:

- Un cluster Amazon EKS Auto Mode con la voce di `AWSServiceRoleForAmazonEKS` accesso creata automaticamente.

- `kubectl`configurato con accesso al cluster.
- La AWS CLI installata e configurata con le autorizzazioni per `aws sts get-caller-identity` e. `aws eks describe-access-entry`
- Un ALB `Ingress` che utilizza l'`alb.ingress.kubernetes.io/auth-type: oidc`annotazione e il Kubernetes a cui fa riferimento. `Secret`

## Fase 1: Identifica il segreto da aggiungere all'elenco delle autorizzazioni

La configurazione OIDC su un ALB è la `Ingress` seguente. Registra il `Secret` nome e lo spazio dei `Ingress` nomi: gli oggetti RBAC devono risiedere nello stesso spazio dei nomi di. `Secret`

```
annotations:
  alb.ingress.kubernetes.io/auth-type: oidc
  alb.ingress.kubernetes.io/auth-idp-oidc: >
    {
      "issuer": "...",
      "authorizationEndpoint": "...",
      "tokenEndpoint": "...",
      "userInfoEndpoint": "...",
      "secretName": "oidc"
    }
```

Il resto di questo esempio utilizza namespace e name. `monitoring` `Secret` `oidc`

## Passaggio 2: applicare un ruolo con ambito di namespace e RoleBinding

Questo modulo concede esattamente un nome `get` in un unico namespace. `Secret` Salva quanto segue come: `eks-managed-oidc-secret-reader.yaml`

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eks-managed-oidc-secret-reader
  namespace: monitoring
rules:
- apiGroups: [""]
  resources: ["secrets"]
  resourceNames: ["oidc"]
  verbs: ["get"]
---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-managed-oidc-secret-reader
  namespace: monitoring
subjects:
- kind: Group
  name: eks:managed
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-managed-oidc-secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Applicalo:

```
kubectl apply -f eks-managed-oidc-secret-reader.yaml
```

Per aggiungere altri `Secret` oggetti all'elenco degli oggetti consentiti in un secondo momento, aggiungi i loro nomi all'`Role's `resourceNames`elenco e riapplica.

## Fase 3: Conferma il ripristino di Ingress

Il controller del load balancer li riconcilia al passaggio successivo (`Ingress`in genere entro pochi minuti). Quando la concessione funziona, l'`secrets "oidc" is forbidden`errore precedente smette di apparire su `Ingress` e il load balancer procede. Se l'errore persiste, ricontrolla il `Role's `namespaceresourceNames`, e l'oggetto. `RoleBinding`

## Rimozione della sovvenzione

> **Important**
>
> Mantieni `Role` la mano `RoleBinding` in posizione per tutto il tempo in cui l'ALB lo `Ingress` utilizza`alb.ingress.kubernetes.io/auth-type: oidc`. La loro rimozione interrompe nuovamente la risoluzione OIDC su. `Ingress`

Se smetti di usare OIDC su, rimuovi la `Ingress` concessione:

```
kubectl delete rolebinding eks-managed-oidc-secret-reader -n monitoring
```

```
kubectl delete role        eks-managed-oidc-secret-reader -n monitoring
```

### Risorse correlate

- the section called "Concedi RBAC ai controller gestiti"— Panoramica concettuale del `eks:managed` gruppo e come definire l'ambito delle sovvenzioni.
- the section called "Voci di accesso"— Voci di accesso ad Amazon EKS.
- Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.

## Crea una classe di archiviazione

Una `StorageClass` in Amazon EKS Auto Mode definisce come i volumi Amazon EBS vengono forniti automaticamente quando le applicazioni richiedono uno storage persistente. In questa pagina viene spiegato come creare e configurare una `StorageClass` che funzioni con Amazon EKS Auto Mode per il provisioning dei volumi EBS.

Configurando una `StorageClass`, è possibile specificare le impostazioni predefinite per i volumi EBS, inclusi il tipo di volume, la crittografia, gli IOPS e altri parametri di storage. Puoi anche configurare l'utilizzo delle `StorageClass` chiavi AWS KMS per la gestione della crittografia.

EKS Auto Mode non crea una `StorageClass` per te. Devi creare una `StorageClass` che faccia riferimento a `ebs.csi.eks.amazonaws.com` per usare la funzionalità di storage di EKS Auto Mode.

Per prima cosa, crea un file denominato `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
allowedTopologies:
- matchLabelExpressions:
  - key: eks.amazonaws.com/compute-type
    values:
    - auto
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
```

```
parameters:
  type: gp3
  encrypted: "true"
```

Poi applica la classe di archiviazione al cluster.

```
kubectl apply -f storage-class.yaml
```

Componenti chiave:

- `provisioner: ebs.csi.eks.amazonaws.com`: usa EKS Auto Mode
- `allowedTopologies`: specificando `matchLabelExpressions` in modo che corrisponda con `eks.amazonaws.com/compute-type:auto`, se i pod richiedono un volume su cui effettuare il provisioning automatico usando Auto Mode, i pod non verranno pianificati su nodi non automatici.
- `volumeBindingMode: WaitForFirstConsumer`: ritarda la creazione del volume fino a quando un pod non ne ha bisogno
- `type: gp3`: specifica il tipo di volume EBS
- `encrypted: "true"`: EBS crittograferà tutti i volumi creati usando la `StorageClass`. EBS userà l'alias chiave predefinito `aws/ebs`. Per ulteriori informazioni, consulta How Amazon EBS encryption works nella Guida per l'utente di Amazon EBS. Questo valore è opzionale ma consigliato.
- `storageclass.kubernetes.io/is-default-class: "true"`: Kubernetes userà questa classe di archiviazione per impostazione predefinita, a meno che non venga specificata una classe di volume diversa su una dichiarazione di volume persistente. Questo valore è facoltativo. Presta attenzione quando imposti questo valore se stai effettuando la migrazione da un controller di storage diverso.

## Usa la chiave KMS autogestita per crittografare i volumi EBS

Per usare una chiave KMS autogestita per crittografare i volumi EBS automatizzati da EKS Auto Mode, bisogna:

1. Creare una chiave KMS gestita dal cliente.
   - Per ulteriori informazioni, consulta Create a symmetric encryption KMS key o How Amazon Elastic Block Store (Amazon EBS) uses KMS nella Guida per gli utenti di KMS.
2. Creare una nuova policy che consenta l'accesso alla chiave KMS.

- Usa la policy IAM di esempio riportata di seguito per creare la policy. Inserisci l'ARN della nuova chiave KMS autogestita. Per ulteriori informazioni, consulta Creazione di ruoli e collegamento di politiche (console) nella Guida per l'utente AWS IAM.

3. Collegare la policy al ruolo del cluster EKS.
   - Usa la AWS console per trovare l'ARN dell'EKS Cluster Role. Le informazioni relative al ruolo sono visibili nella sezione Panoramica. Per ulteriori informazioni, consulta the section called "Ruolo IAM del cluster".
4. Aggiornare la `StorageClass` per fare riferimento all'ID della chiave KMS nel campo `parameters.kmsKeyId`.

Esempio di policy IAM KMS autogestita

Aggiorna i valori seguenti nella policy riportata di seguito:

- `<account-id>`— L'ID AWS del tuo account, ad esempio `111122223333`
- `<aws-region>`— La AWS regione del cluster, ad esempio `us-west-2`

```
{
  "Version":"2012-10-17",
  "Id": "key-auto-policy-3",
  "Statement": [
      {
          "Sid": "Enable IAM User Permissions",
          "Effect": "Allow",
          "Principal": {
              "AWS": "arn:aws:iam::123456789012:root"
          },
          "Action": "kms:*",
          "Resource": "*"
      },
      {
        "Sid": "Allow access through EBS for all principals in the account that are
 authorized to use EBS",
        "Effect": "Allow",
        "Principal": {
            "AWS": "*"
        },
        "Action": [
            "kms:Encrypt",
```

```
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:CallerAccount": "123456789012",
                "kms:ViaService": "ec2.us-east-1.amazonaws.com"
            }
        }
    }
  ]
}
```

Esempio di **`StorageClass`** KMS gestita dal cliente

```
parameters:
  type: gp3
  encrypted: "true"
  kmsKeyId: <custom-key-arn>
```

## Guida di riferimento per i parametri **`StorageClass`**

Per informazioni generali sulle risorse `StorageClass` di Kubernetes, consulta Storage Classes nella documentazione di Kubernetes.

La `parameters` sezione della `StorageClass` risorsa è specifica per AWS. Utilizza la tabella seguente per esaminare le opzioni disponibili.

| Parameters | Valori | Predefinita | Description |
|---|---|---|---|
| «csi.storage.k8s. io/ fstype» | xfs, ext2, ext3, ext4 | ext4 | Tipo di file system che verrà formattat o al momento della creazione del volume. Questo parametro prevede la distinzio |

| Parameters | Valori | Predefinita | Description |
| --- | --- | --- | --- |
| | | | ne tra maiuscole e minuscole. |
| "type" | io1, io2, gp2, gp3, sc1, st1, standard, sbp1, sbg1 | gp3 | Tipo di volume EBS. |
| "iopsPerGB" | | | I/O operazioni al secondo per GiB. Possono essere specificate per i volumi IO1, IO2 e GP3. |

| Parameters | Valori | Predefinita | Description |
| --- | --- | --- | --- |
| «consentireAutoIOPSPerGBIncrease» | true, false | false | Quando`"true"`, il driver CSI aumenta gli IOPS per un volume troppo basso per rientrare nell'intervallo IOPS supportato da. `iopsPerGB * <volume size>` AWS Ciò consente al provisioning dinamico di avere sempre esito positivo, anche quando l'utente specifica una capacità PVC o un valore `iopsPerGB` troppo bassi. D'altra parte, ciò potrebbe comportare costi aggiuntivi, poiché tali volumi hanno un IOPS superiore a quello richiesto in `iopsPerGB` . |
| "iops" | | | I/O operazioni al secondo. Possono essere specificate per i volumi IO1, IO2 e GP3. |
| "throughput" | | 125 | Throughput in entrata MiB/s. Efficace solo quando è specificato il tipo di volume gp3. |

| Parameters | Valori | Predefinita | Description |
| --- | --- | --- | --- |
| "encrypted" | true, false | false | Se il volume deve essere crittografato o no. I valori validi sono "true" o "false". |
| "blockExpress" | true, false | false | Permette la creazione di Volumi io2 Block Express. |
| «kmKeyId» | | | L'ARN completo della chiave da usare per la crittografia del volume. Se non specificato, AWS utilizzerà la chiave KMS predefinita per la regione in cui si trova il volume. Si tratta di una chiave generata automaticamente chiamata `/aws/ebs` se non viene modificat a. |
| "blockSize" | | | La dimensione del blocco da utilizzar e per la formattaz ione del filesystem sottostante. Supportat a solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4` o `xfs`. |

| Parameters | Valori | Predefinita | Description |
| --- | --- | --- | --- |
| "inodeSize" | | | La dimensione dell'inode da utilizzare per la formattazione del filesystem sottostante. Supportata solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4` o `xfs`. |
| «bytePerInode» | | | `bytes-per-inode` da utilizzare per la formattazione del filesystem sottostante. Supportata solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4`. |
| «numeroOfInodes» | | | `number-of-inodes` da utilizzare per la formattazione del filesystem sottostante. Supportata solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4`. |

| Parameters | Valori | Predefinita | Description |
| --- | --- | --- | --- |
| «ext4BigAlloc» | true, false | false | Modifica il filesyste m `ext4` per usare l'allocazione di blocchi in cluster abilitando l'opzione di formattazione `bigalloc`. Attenzion e: `bigalloc` potrebbe non essere completamente supportato dal kernel Linux del nodo. |
| «ClusterSizeext4» | | | La dimensione del cluster da usare per formattare un filesystem `ext4` quando la funzionalità `bigalloc` è abilitata . Nota: il parametro `ext4BigAlloc` deve essere impostato su true |

Per ulteriori informazioni, consulta AWS EBS CSI Driver su. GitHub

## Considerazioni

> **Note**
>
> È possibile distribuire carichi di lavoro solo in base alla modalità EKS Auto Mode sui nodi EKS Auto Mode StorageClasses . Se hai un cluster con tipi misti di nodi, devi configurare i carichi di lavoro in modo che vengano eseguiti solo sui nodi EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Controllare l'implementazione".

La funzionalità di storage a blocchi di EKS Auto Mode è diversa da quella del driver CSI EBS.

- Provisioning statico
  - Se desideri utilizzare volumi EBS creati esternamente con EKS Auto Mode, devi aggiungere manualmente un AWS tag con la chiave `eks:eks-cluster-name` e il valore del nome del cluster.
- Taint startup del nodo
  - Non è possibile usare la funzionalità taint startup del nodo per impedire la pianificazione dei pod prima che la capacità di storage sia pronta
- Tag personalizzati su volumi con provisioning dinamico
  - Non è possibile usare il flag CLI extra-tag per configurare tag personalizzati su volumi EBS con provisioning dinamico
  - Puoi usare l'assegnazione di tag `StorageClass` per aggiungere tag personalizzati. EKS Auto Mode aggiungerà tag alle risorse associate. AWS Dovrai aggiornare il ruolo del IAM cluster per i tag personalizzati. Per ulteriori informazioni, consulta the section called "Personalizzato AWS tag per le risorse EKS Auto".
- Metriche sulle prestazioni dettagliate di EBS
  - Non puoi accedere alle metriche di Prometheus per le prestazioni dettagliate di EBS

## Installa il componente aggiuntivo CSI Snapshot Controller

EKS Auto Mode è compatibile con il componente aggiuntivo CSI Snapshot Controller di Amazon EKS.

AWS suggerisce di configurare questo componente aggiuntivo per l'esecuzione nel pool di `system` nodi integrato.

Per ulteriori informazioni, consulta:

- the section called "Eseguire componenti aggiuntivi critici"
- the section called "Riesamina i pool di nodi integrati"
- the section called "Controller snapshot CSI"

Per installare il controller di snapshot nel pool di nodi di sistema

1. Apri il tuo cluster EKS nella console AWS

2. Dalla Add-onsscheda, seleziona Ottieni altri componenti aggiuntivi
3. Seleziona il controller CSI Snapshot e poi Avanti
4. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona Impostazioni di configurazione opzionali per visualizzare lo Add-on schema di configurazione
   a. Inserisci il seguente yaml per associare il controller di snapshot al pool di nodi `system`. Il controller di snapshot include una tolleranza per il taint `CriticalAddonsOnly`.

```
{
        "nodeSelector": {
            "karpenter.sh/nodepool": "system"
        }
}
```

   b. Seleziona Avanti
5. Esamina la configurazione del componente aggiuntivo, poi seleziona Crea

## Disabilita modalità automatica di EKS

Puoi disabilita modalità automatica di EKS su un cluster EKS esistente. Si tratta di un'operazione distruttiva.

- EKS interromperà tutte le istanze EC2 gestite dalla modalità automatica di EKS.
- EKS eliminerà tutti i bilanciatori del carico gestiti dalla modalità automatica di EKS.
- EKS non eliminerà i volumi EBS forniti dalla modalità automatica di EKS.

modalità automatica di EKS è progettato per gestire completamente le risorse che crea. Gli interventi manuali potrebbero impedire alla modalità automatica di EKS di ripulire completamente tali risorse quando è disabilitata. Ad esempio, se hai fatto un riferimento a un gruppo di sicurezza gestito da regole esterne del gruppo di sicurezza e ti dimentica di rimuovere quel riferimento prima di disabilitare modalità automatica di EKS per un cluster, il gruppo di sicurezza gestito verrà divulgato (non verrà eliminato). Le fasi seguenti descrivono come rimuovere un gruppo di sicurezza trapelato se ciò dovesse accadere.

### Disattiva la modalità automatica EKS (AWS Console)

1. Apri la pagina di panoramica del cluster in Console di gestione AWS.
2. In modalità automatica di EKS, seleziona Gestisci

3. Attiva modalità automatica di EKS su `off`.

Se un gruppo di sicurezza gestito non viene eliminato al termine di questo processo, è possibile eliminarlo manualmente utilizzando le istruzioni in Eliminare un gruppo di sicurezza.

### Disabilita la modalità automatica EKS (AWS CLI)

Utilizza il seguente comando per disabilitare modalità automatica di EKS su un cluster esistente.

È necessario che la `aws` CLI sia installata ed essere loggati con autorizzazioni sufficienti per gestire i cluster EKS. Per ulteriori informazioni, consulta Configurazione.

> **Note**
>
> Le funzionalità di calcolo, archiviazione a blocchi e bilanciamento del carico devono essere tutte abilitate o disabilitate nella stessa richiesta.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=false \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": false}}' \
 --storage-config '{"blockStorage":{"enabled": false}}'
```

Puoi verificare se non è stato possibile eliminare un gruppo di sicurezza della modalità automatica di EKS divulgato dopo aver disabilitato modalità automatica di EKS nel modo seguente:

```
aws ec2 describe-security-groups \
    --filters Name=tag:eks:eks-cluster-name,Values=<cluster-Name> Name=tag-
key,Values=ingress.eks.amazonaws.com/resource,service.eks.amazonaws.com/resource --
query "SecurityGroups[*].[GroupName]"
```

Poi, per eliminare il gruppo di sicurezza:

```
aws ec2 delete-security-group --group-name=<sg-name>
```

## Aggiornare la versione Kubernetes di un cluster di Amazon EKS Auto Mode

Questo argomento ti spiega come aggiornare la versione di Kubernetes per il cluster Auto Mode. Auto Mode semplifica il processo di aggiornamento delle versioni gestendo il coordinamento degli

aggiornamenti del piano di controllo con la sostituzione dei nodi, mantenendo al contempo la disponibilità del carico di lavoro attraverso i budget relativi all'interruzione dei pod.

Quando si aggiorna un cluster Auto Mode, molti componenti che tradizionalmente richiedevano aggiornamenti manuali vengono ora gestiti come parte del servizio. Comprendere gli aspetti automatizzati del processo di aggiornamento e le proprie responsabilità aiuta a garantire una transizione fluida delle versioni per il cluster.

## Informazioni sugli aggiornamenti con EKS Auto Mode

Dopo che hai avviato un aggiornamento del piano di controllo, EKS Auto Mode aggiornerà i nodi del cluster. Alla scadenza dei nodi, EKS Auto Mode li sostituirà con nuovi nodi. I nuovi nodi hanno la nuova versione di Kubernetes corrispondente. EKS Auto Mode rispetta i budget relativi alle interruzioni dei pod durante l'aggiornamento dei nodi.

Inoltre, non dovrai più aggiornare componenti come:

- CNI di Amazon VPC
- Controller del load balancerAWS
- CoreDNS
- `kube-proxy`
- Karpenter
- AWS Driver CSI per EBS

EKS Auto Mode sostituisce questi componenti con funzionalità di servizio.

L'utente è ancora responsabile dell'aggiornamento di:

- App e carichi di lavoro implementati nel cluster
- Componenti aggiuntivi e controller autogestiti
- Componenti aggiuntivi di Amazon EKS
  - Informazioni su come the section called "Aggiornamento di un componente aggiuntivo"

Informazioni su Best Practices for Cluster Upgrades

### Avviare l'aggiornamento del cluster

Per avviare un aggiornamento del cluster, consulta the section called "Aggiornamento della versione di Kubernetes".

## Attivazione o disattivazione Built-in NodePools

La modalità automatica EKS ne ha due integrate NodePools. Puoi abilitarli o disabilitarli NodePools utilizzando la AWS console, la CLI o l'API.

### Built-in NodePool Riferimento

- `system`
  - Questo NodePool ha una `CriticalAddonsOnly` macchia. Molti componenti aggiuntivi EKS, come CoredNS, tollerano questo taint. Utilizza questo pool di nodi di sistema per separare le applicazioni critiche per il cluster.
  - Supporta entrambe le architetture `amd64` e `arm64`.
- `general-purpose`
  - Ciò NodePool fornisce supporto per l'avvio di nodi per carichi di lavoro generici nel cluster.
  - Utilizza solo l'architettura `amd64`.

Entrambi integrati: NodePools

- Usa l'EKS predefinito NodeClass
- Utilizza solo la capacità EC2 su richiesta
- Utilizza le famiglie di istanze EC2 C, M e R
- Richiedere le istanze EC2 di generazione 5 o successive

> **Note**
>
> NodePool È necessario abilitare almeno un sistema integrato per consentire a EKS di fornire il «valore predefinito» NodeClass. Se disabiliti tutte le funzionalità integrate NodePools, dovrai crearne una personalizzata NodeClass e configurarne una NodePool per utilizzarla. Per ulteriori informazioni su NodeClasses, consultathe section called "Creazione di una classe di nodi".

> **Important**
>
> Quando rimuovi un NodePool nome integrato da`computeConfig.nodePools`, la risorsa NodePool Kubernetes corrispondente viene eliminata dal cluster. Tutti i nodi da esso gestiti NodePool vengono svuotati e terminati. Quando si aggiunge nuovamente un NodePool nome integrato a`computeConfig.nodePools`, la NodePool risorsa corrispondente viene ricreata nel cluster. Questo comportamento è importante quando si pianificano le modifiche alla capacità, perché Karpenter rileva NodePool le risorse tramite l'API Kubernetes e interrompe la pianificazione dei nodi da eliminare. NodePools

## Procedura

### Prerequisiti

- La versione più recente dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.
  - Accedi alla CLI con autorizzazioni IAM sufficienti per creare AWS risorse tra cui politiche IAM, ruoli IAM e cluster EKS.

### Abilita con AWS CLI

Usa il seguente comando per abilitare entrambi i built-in NodePools:

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
    "nodeRoleArn": "<node-role-arn>",
    "nodePools": ["general-purpose", "system"],
    "enabled": true
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}
  }' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

È possibile modificare il comando per abilitare selettivamente il NodePools.

Disabilita con AWS CLI

Usa il seguente comando per disabilitare entrambi i componenti integrati NodePools:

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
  "enabled": true,
  "nodePools": []
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}}' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

## Controllare se un carico di lavoro viene implementato sui nodi della modalità automatica di EKS

Quando si eseguono carichi di lavoro in un cluster EKS con la modalità automatica di EKS, potresti dover controllare se carichi di lavoro specifici vengono eseguiti su nodella modalità automatica di EKS o altri tipi di elaborazione. Questo argomento descrive come utilizzare i selettori di nodi e le regole di affinità per garantire che i carichi di lavoro siano pianificati sull'infrastruttura di elaborazione prevista.

Gli esempi in questo argomento mostrano come utilizzare l'etichetta `eks.amazonaws.com/compute-type` per richiedere o impedire l'implementazione del carico di lavoro sui nodi della modalità automatica di EKS. Ciò è particolarmente utile nei cluster a modalità mista in cui è in esecuzione sia modalità automatica di EKS che altri tipi di elaborazione, come i provisioner Karpenter autogestiti o i gruppi di nodi gestiti da EKS.

I nodi della modalità automatica di EKS hanno impostato il valore dell'etichetta da `eks.amazonaws.com/compute-type` a `auto`. Puoi utilizzare questa etichetta per controllare se un carico di lavoro viene implementato nei nodi gestiti dalla modalità automatica di EKS.

## Richiedere che un carico di lavoro sia distribuito sui nodi EKS Auto Mode

> **Note**
>
> Questo valore `nodeSelector` non è richiesto per la modalità automatica di EKS. Questo valore `nodeSelector` è rilevante solo se si esegue un cluster in modalità mista, con tipi di nodi non gestiti dalla modalità automatica di EKS. Ad esempio, puoi avere una capacità di elaborazione statica implementata nel cluster con o gruppi di nodi gestiti da EKS e avere una capacità di elaborazione dinamica gestita dalla modalità automatica di EKS.

Puoi aggiungere questo `nodeSelector` alle implementazioni o ad altri carichi di lavoro per richiedere che Kubernetes li pianifichi sui nodi della modalità automatica di EKS.

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

## Richiede che un carico di lavoro non sia distribuito sui nodi EKS Auto Mode

Puoi aggiungere questo `nodeAffinity` alle implementazioni o ad altri carichi di lavoro per richiedere che Kubernetes non li pianifichi sui nodi della modalità automatica di EKS.

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - auto
```

## Scegli come target uno specifico NodePool

Un cluster Amazon EKS Auto Mode può averne più di uno NodePool. Potresti volere che un carico di lavoro venga eseguito solo sui nodi di un determinato NodePool nodo. A tale scopo, i carichi di lavoro possono corrispondere a una delle seguenti etichette:

- Un'etichetta personalizzata definita in. NodePool  Quando aggiungi etichette `spec.template.metadata.labels` in a NodePool, Amazon EKS Auto Mode applica tali etichette a tutti i nodi di cui esegue il NodePool provisioning. Le etichette personalizzate sono l'approccio consigliato. Un'etichetta personalizzata lega il carico di lavoro a una funzionalità, ad esempio`workload-class: gpu-inference`, anziché a un NodePool nome specifico.
- L'**`karpenter.sh/nodepool`**etichetta ben nota.  Amazon EKS Auto Mode applica questa etichetta a tutti i nodi di cui esegue il provisioning, utilizzando il nome di NodePool come valore. Usa questa etichetta quando non hai definito un'etichetta personalizzata e vuoi impostarne una NodePool per nome.

### Scegli come target un' NodePool etichetta personalizzata

Innanzitutto, definisci un'etichetta su NodePool.

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  template:
    metadata:
      labels:
        workload-class: gpu-inference
```

Per ulteriori informazioni sulla NodePool configurazione, vederethe section called “Creazione di pool di nodi”.

Quindi, abbina l'etichetta del carico di lavoro.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inference
spec:
```

```
  template:
    spec:
      nodeSelector:
        workload-class: gpu-inference
```

I pod di questa pianificazione di distribuzione si limitano ai nodi di cui dispone. `gpu` NodePool I carichi di lavoro senza questa impostazione `nodeSelector` continuano a essere eseguiti sulle impostazioni predefinite. NodePool

Scegli come target a NodePool per nome

> **NodePool le modifiche al nome influiscono sulla pianificazione**
>
> Se si rinomina o si elimina la destinazione, se si aggiunge un nome a un carico di lavoro in base al nome`Pending`, i pod rimangono invariati. NodePool Usa un'etichetta di funzionalità personalizzata per qualsiasi carico di lavoro che prevedi possa durare più a lungo di una singola definizione. NodePool

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      nodeSelector:
        karpenter.sh/nodepool: <your-nodepool-name>
```

Per ulteriori informazioni su come Karpenter abbina i pod NodePools, consulta Scheduling nella documentazione di Karpenter.

## Eseguire componenti aggiuntivi critici su istanze dedicate

Questo argomento illustra come implementare un carico di lavoro con una tolleranza `CriticalAddonsOnly` in modo che la modalità automatica EK lo pianifichi nel pool di nodi `system`.

Il pool di nodi integrato `system` della modalità automatica EKS è progettato per eseguire componenti aggiuntivi critici su istanze dedicate. Questa segregazione garantisce che i componenti essenziali dispongano di risorse dedicate e siano isolati dai carichi di lavoro generali, migliorando la stabilità e le prestazioni complessive del cluster.

Questa guida illustra come implementare componenti aggiuntivi nel pool di nodi `system` utilizzando la tolleranza `CriticalAddonsOnly` e i selettori dei nodi adeguati. Seguendo questa procedura, è possibile garantire che le applicazioni critiche siano pianificate sui nodi `system` dedicati, sfruttando i vantaggi dell'isolamento e dell'allocazione delle risorse offerti dalla struttura specializzata dei pool di nodi della modalità automatica EKS.

La modalità automatica EKS dispone di due pool di nodi integrati: `general-purpose` e `system`. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

Lo scopo del pool di nodi `system` è segregare i componenti aggiuntivi critici su nodi diversi. I nodi con provisioning del pool di nodi `system` hanno un taint Kubernetes `CriticalAddonsOnly`. Kubernetes pianificherà i pod su questi nodi solo se dispongono di una tolleranza corrispondente. Per ulteriori informazioni, consultare Taints and Tolerations nella documentazione di Kubernetes.

## Prerequisiti

- Cluster di modalità automatica EKS con il pool di nodi integrato `system` abilitato. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati"
- `kubectl` installato e configurato. Per ulteriori informazioni, consulta Configurazione.

## Procedura

Rivedere l'esempio yaml riportato di seguito. Tenere presente le configurazioni seguenti:

- `nodeSelector`: questa configurazione associa il carico di lavoro al pool di nodi integrato `system`. Questo pool di nodi deve essere abilitato con l'API AWS. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".
- `tolerations`: questa tolleranza supera il taint `CriticalAddonsOnly` sui nodi del pool di nodi `system`.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
```

```
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      nodeSelector:
        karpenter.sh/nodepool: system
      tolerations:
      - key: "CriticalAddonsOnly"
        operator: "Exists"
      containers:
      - name: app
        image: nginx:latest
        resources:
          requests:
            cpu: "500m"
            memory: "512Mi"
```

Per aggiornare un carico di lavoro affinché venga eseguito sul pool di nodi `system` è necessario:

1. Aggiornare il carico di lavoro esistenti affinché aggiunga le seguenti configurazioni descritte in precedenza:
   - `nodeSelector`
   - `tolerations`
2. Implementare il carico di lavoro aggiornato nel cluster con `kubectl apply`.

Una volta aggiornato, il carico di lavoro sarà eseguito su nodi dedicati.

## Usa le policy di rete con la modalità automatica di EKS

### Panoramica di

Man mano che i clienti scalano i propri ambienti applicativi utilizzando EKS, l'isolamento del traffico di rete diventa sempre più fondamentale per prevenire l'accesso non autorizzato alle risorse all'interno e all'esterno del cluster. Ciò è particolarmente importante in un ambiente multi-tenant con più carichi di lavoro non correlati eseguiti fianco a fianco nel cluster. Le policy di rete Kubernetes consentono di migliorare il livello di sicurezza della rete per i carichi di lavoro Kubernetes e le relative integrazioni con gli endpoint esterni del cluster. EKS Auto Mode supporta diversi tipi di policy di rete.

### Isolamento di livello 3 e 4

Le policy di rete Kubernetes standard operano ai livelli 3 e 4 del modello di rete OSI e consentono di controllare il flusso di traffico a livello di indirizzo IP o porta all'interno del cluster Amazon EKS.

#### Casi d'uso

- Segmenta il traffico di rete tra i carichi di lavoro per garantire che solo le applicazioni correlate possano comunicare tra loro.
- Isola i tenant a livello di namespace utilizzando politiche per imporre la separazione della rete.

### DNS-based applicazione

I clienti in genere implementano carichi di lavoro in EKS che fanno parte di un ambiente distribuito più ampio, alcuni dei quali devono comunicare con sistemi e servizi esterni al cluster (traffico in direzione nord). Questi sistemi e servizi possono essere nel AWS cloud o completamente all'esterno. AWS Le policy basate sul Domain Name System (DNS) consentono di rafforzare il livello di sicurezza adottando un approccio più stabile e prevedibile per prevenire l'accesso non autorizzato dai pod alle risorse o agli endpoint esterni del cluster. Questo meccanismo elimina la necessità di tracciare e consentire manualmente un elenco di indirizzi IP specifici. Proteggendo le risorse con un DNS-based approccio, si ha anche una maggiore flessibilità per aggiornare l'infrastruttura esterna senza dover allentare il livello di sicurezza o modificare le politiche di rete a causa delle modifiche ai server e agli host a monte. È possibile filtrare il traffico in uscita verso endpoint esterni utilizzando un nome di dominio completo (FQDN) o uno schema corrispondente per un nome di dominio DNS. Ciò offre la maggiore flessibilità di estendere l'accesso a più sottodomini associati a un particolare endpoint esterno del cluster.

#### Casi d'uso

- Stabilisci un DNS-based approccio standardizzato per filtrare l'accesso da un ambiente Kubernetes agli endpoint esterni del cluster.
- Accesso sicuro ai servizi in un ambiente multi-tenant. AWS
- Gestisci l'accesso alla rete dai pod ai carichi di lavoro on-premise nei tuoi ambienti cloud ibridi.

### Regole amministrative (o con ambito di cluster)

In alcuni casi, come negli scenari multi-tenant, i clienti potrebbero dover applicare uno standard di sicurezza di rete applicabile all'intero cluster. Invece di definire e mantenere ripetutamente una

policy distinta per ogni namespace, è possibile utilizzare un'unica policy per gestire centralmente i controlli di accesso alla rete per i diversi carichi di lavoro nel cluster, indipendentemente dal rispettivo namespace. Questi tipi di policy consentono di estendere l'ambito di applicazione delle regole di filtro di rete applicate a livello 3, livello 4 e quando si utilizzano le regole DNS.

Casi d'uso

- Gestisci centralmente i controlli di accesso alla rete per tutti (o un sottoinsieme di) carichi di lavoro nel tuo cluster EKS.
- Definisci un livello di sicurezza di rete predefinito in tutto il cluster.
- Estendi gli standard di sicurezza organizzativi all'ambito del cluster in un modo più efficiente dal punto di vista operativo.

## Nozioni di base

Prerequisiti

- Un cluster Amazon EKS con la modalità automatica di EKS abilitata
- kubectl configurato per connettersi al cluster

Fase 1: abilitare il controller della policy di rete

Per utilizzare le policy di rete con EKS Auto Mode, devi prima abilitare il Network Policy Controller applicando a ConfigMap al tuo cluster.

1. Crea un file denominato `enable-network-policy.yaml` con i seguenti contenuti:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

2. Applicalo ConfigMap al tuo cluster:

```
kubectl apply -f enable-network-policy.yaml
```

Passaggio 2: creare e testare le politiche di rete

Il cluster della modalità automatica di EKS è ora configurato per supportare le policy di rete di Kubernetes. Puoi testarlo con la the section called "Demo policy Stars".

Passaggio 3: modifica della configurazione del Network Policy Agent nella classe Node (opzionale)

Facoltativamente, è possibile creare una nuova classe di nodi per modificare il comportamento predefinito del Network Policy Agent sui nodi o abilitare la registrazione degli eventi di Network Policy. A tale scopo, seguire queste fasi:

1. Crea o modifica un file YAML di classe del nodo (ad esempio, `nodeclass-network-policy.yaml`) con i seguenti contenuti:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: network-policy-config
spec:
  # Optional: Changes default network policy behavior
  networkPolicy: DefaultAllow
  # Optional: Enables logging for network policy events
  networkPolicyEventLogs: Enabled
  # Include other Node Class configurations as needed
```

2. Applica la configurazione di classe del nodo al cluster:

```
kubectl apply -f nodeclass-network-policy.yaml
```

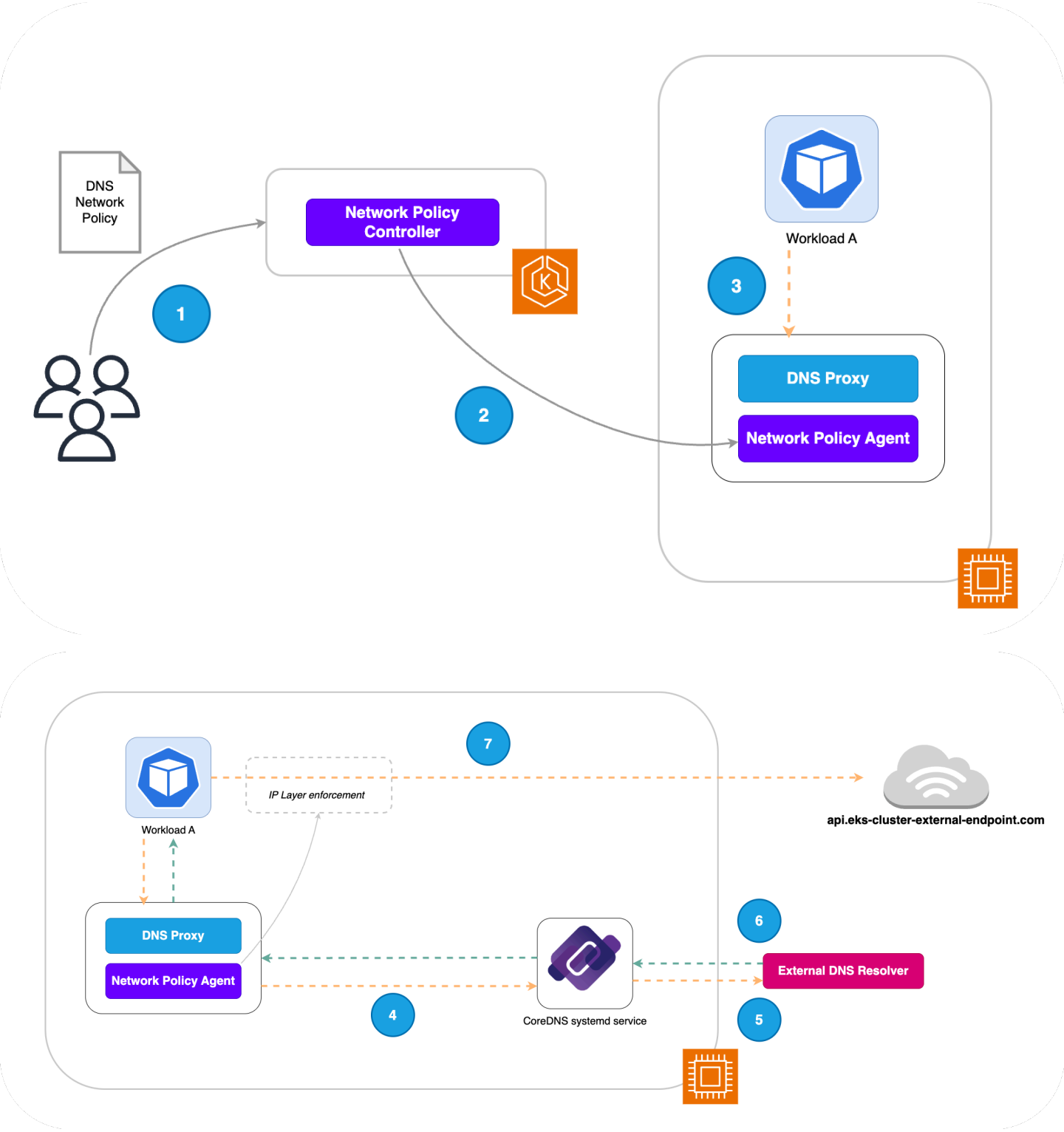
3. Verifica che la classe del nodo sia stata creata:

```
kubectl get nodeclass network-policy-config
```

4. Aggiorna il pool di nodi per utilizzare questa classe di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

# Come funziona?

## DNS-based politica di rete




1. Il team della piattaforma applica una DNS-based policy al cluster EKS.

2. Il Network Policy Controller è responsabile del monitoraggio della creazione delle politiche all'interno del cluster e quindi della riconciliazione degli endpoint delle politiche. In questo caso d'uso, il controller delle policy di rete indica all'agente del nodo di filtrare le richieste DNS in base ai domini consentiti nella policy creata. I nomi di dominio sono elencati nell'elenco consentito utilizzando l'FQDN o un nome di dominio che corrisponde a uno schema definito nella configurazione delle risorse Kubernetes.
3. Il carico di lavoro A tenta di risolvere l'IP per un endpoint esterno al cluster. La richiesta DNS passa innanzitutto attraverso un proxy che filtra tali richieste in base all'elenco di autorizzazioni applicato tramite la politica di rete.
4. Dopo che la richiesta DNS è passata attraverso l'elenco dei filtri DNS consentiti, il proxy la inoltra a CoreDNS,
5. CoreDNS a sua volta invia la richiesta al resolver DNS esterno (Amazon Route 53 Resolver) per ottenere l'elenco degli indirizzi IP dietro il nome di dominio.
6. Gli IP risolti con TTL vengono restituiti nella risposta alla richiesta DNS. Questi IP vengono quindi scritti in una mappa eBPF che viene utilizzata nella fase successiva per l'applicazione del livello IP.
7. Le sonde eBPF collegate all'interfaccia Pod veth filtreranno quindi il traffico in uscita dal carico di lavoro A all'endpoint esterno del cluster in base alle regole in vigore. Ciò garantisce che i pod possano inviare solo traffico esterno al cluster agli IP dei domini consentiti elencati. La validità di questi IP si basa sul TTL recuperato dal resolver DNS esterno (Amazon Route 53 Resolver).

### Utilizzo dell'Application Network Policy

`ApplicationNetworkPolicy`Combina le funzionalità delle policy di rete standard di Kubernetes con il filtro basato su DNS a livello di namespace utilizzando un'unica Custom Resource Definition (CRD). Pertanto, può essere utilizzato per: `ApplicationNetworkPolicy`

1. Definire le restrizioni ai livelli 3 e 4 dello stack di rete utilizzando blocchi IP e numeri di porta.
2. Definizione di regole che operano al livello 7 dello stack di rete e che consentono di filtrare il traffico in base agli FQDN.

> **Important**
>
> Le regole basate sul DNS definite utilizzando il `ApplicationNetworkPolicy` sono applicabili solo ai carichi di lavoro eseguiti nelle istanze EKS Auto EC2. Mode-

launched `ApplicationNetworkPolicy`supporta tutti i campi dello standard Kubernetes`NetworkPolicy`, con un filtro FQDN aggiuntivo per le regole di uscita.

> **Warning**
>
> Non utilizzare lo stesso nome per un `ApplicationNetworkPolicy` e all'interno dello stesso namespace. `NetworkPolicy` Se i nomi si scontrano, gli `PolicyEndpoints` oggetti risultanti potrebbero non riflettere correttamente nessuna delle due politiche. Entrambe le risorse vengono accettate senza errori, il che rende difficile la diagnosi del problema.
> Per risolvere un conflitto di denominazione, rinomina una `ApplicationNetworkPolicy` o le altre `NetworkPolicy` in modo che siano univoci all'interno dello spazio dei nomi, quindi verifica che gli `PolicyEndpoints` oggetti corrispondenti siano aggiornati correttamente.

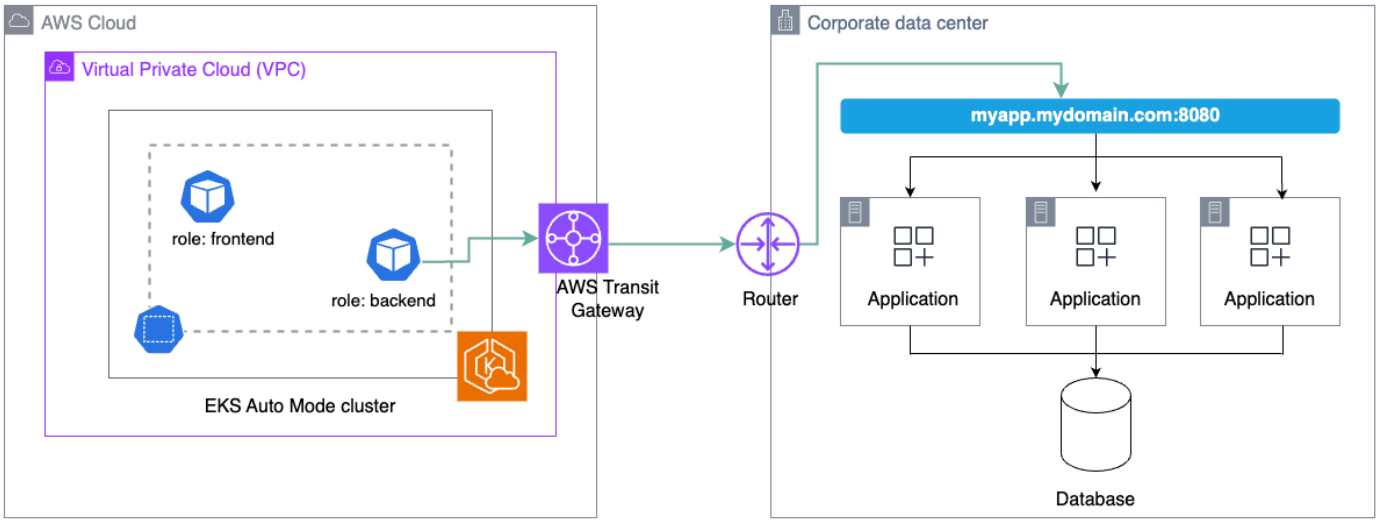
Esempio

Nel cluster EKS Auto Mode è presente un carico di lavoro che deve comunicare con un'applicazione locale che si trova dietro un load balancer con un nome DNS. È possibile ottenere ciò utilizzando la seguente politica di rete:

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ApplicationNetworkPolicy
metadata:
  name: my-onprem-app-egress
  namespace: galaxy
spec:
  podSelector:
    matchLabels:
      role: backend
  policyTypes:
  - Egress
  egress:
  - to:
    - ipBlock: { cidr: 10.100.0.10/32 }
    ports:
    - protocol: TCP
      port: 53
    - protocol: UDP
      port: 53
```

```
  - to:
    - domainNames:
      - "myapp.mydomain.com"
    ports:
    - protocol: TCP
      port: 8080
```

A livello di rete Kubernetes, ciò consentirebbe l'uscita da qualsiasi pod nel namespace «galaxy» etichettato con l'etichetta `role: backend` per connettersi al nome di dominio myapp.mydomain.com sulla porta TCP 8080 e a CoreDNS sulla porta 53. Inoltre, è necessario configurare la connettività di rete per il traffico in uscita dal VPC al data center aziendale.




Determinazione dell'indirizzo IP CoreDNS

Affinché l'applicazione risolva il DNS, deve essere autorizzata a comunicare con CoreDNS, che viene eseguito localmente su ogni istanza di EKS Auto Mode. L'indirizzo IP CoreDNS è derivato dall'intervallo Service CIDR configurato per il cluster e rimane fisso per tutta la durata del cluster. È quindi possibile determinarlo una volta e farvi riferimento direttamente nelle politiche di rete.

1. Recupera il Service CIDR per il tuo cluster.

   Per i cluster IPv4:

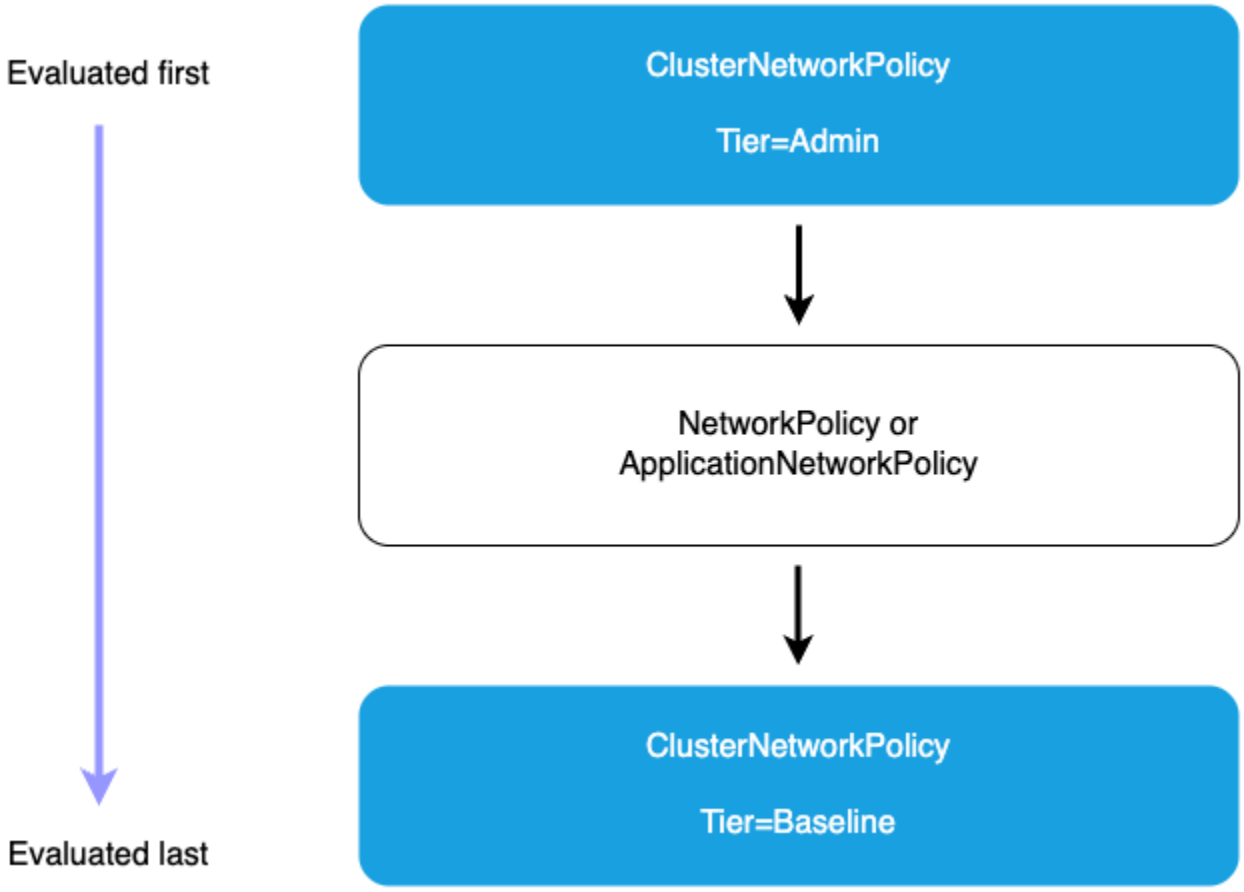
   ```
   aws eks describe-cluster --name my-cluster --query
    'cluster.kubernetesNetworkConfig.serviceIpv4Cidr' --output text
   ```

   Per i cluster IPv6:

```
aws eks describe-cluster --name my-cluster --query
 'cluster.kubernetesNetworkConfig.serviceIpv6Cidr' --output text
```

2. Ricava l'indirizzo IP CoreDNS dal Service CIDR:
   - IPv4: utilizza l'indirizzo del CIDR. `.10` Ad esempio `10.100.0.0/16` diventa `10.100.0.10/32`.
   - IPv6: aggiungi a all'indirizzo di rete. Ad esempio `fd12:3456:789a::/108` diventa `fd12:3456:789a::a/128`.

Politica di rete dell'amministratore (o del cluster)




Utilizzo della Cluster Network Policy

Quando si utilizza a`ClusterNetworkPolicy`, le policy del livello di amministrazione vengono valutate per prime e non possono essere sovrascritte. Una volta valutate le politiche del livello di amministrazione, vengono utilizzate le politiche standard con ambito di namespace per eseguire le regole di segmentazione della rete applicate. Ciò può essere ottenuto utilizzando uno o.

`ApplicationNetworkPolicy` `NetworkPolicy` Infine, verranno applicate le regole del livello di base che definiscono le restrizioni di rete predefinite per i carichi di lavoro del cluster. Queste regole del livello di base possono essere sostituite dalle policy con ambito di namespace, se necessario.

Esempio

Nel cluster è presente un'applicazione che si desidera isolare dai carichi di lavoro degli altri tenant. È possibile bloccare esplicitamente il traffico del cluster proveniente da altri namespace per impedire l'accesso di rete al namespace sensibile del carico di lavoro.

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ClusterNetworkPolicy
metadata:
  name: protect-sensitive-workload
spec:
  tier: Admin
  priority: 10
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: earth
  ingress:
    - action: Deny
      from:
      - namespaces:
          matchLabels: {} # Match all namespaces.
      name: select-all-deny-all
```

## Considerazioni

Comprendi l'ordine di valutazione delle policy

Le funzionalità delle politiche di rete supportate in EKS vengono valutate in un ordine specifico per garantire una gestione del traffico prevedibile e sicura. Pertanto, è importante comprendere il flusso di valutazione per progettare un sistema di sicurezza di rete efficace per il proprio ambiente.

1. Criteri di livello amministrativo (valutati per primi): tutti i livelli di amministrazione ClusterNetworkPolicies vengono valutati prima di qualsiasi altra politica. All'interno del livello Amministratore, le politiche vengono elaborate in ordine di priorità (prima il numero con la priorità più bassa). Il tipo di azione determina cosa succede dopo.

- Azione di rifiuto (precedenza massima): quando una politica di amministrazione con un'azione Nega corrisponde al traffico, tale traffico viene immediatamente bloccato indipendentemente da qualsiasi altra politica. Non vengono elaborate ClusterNetworkPolicy ulteriori NetworkPolicy regole. Ciò garantisce che i controlli di sicurezza a livello di organizzazione non possano essere sostituiti da policy a livello di namespace.
- Consenti azione: dopo aver valutato le regole di rifiuto, le politiche di amministrazione con azioni Consenti vengono elaborate in ordine di priorità (prima il numero con la priorità più bassa). Quando un'azione Consenti corrisponde, il traffico viene accettato e non viene effettuata alcuna ulteriore valutazione della policy. Queste policy possono concedere l'accesso a più namespace in base a selettori di etichette, fornendo un controllo centralizzato su quali carichi di lavoro possono accedere a risorse specifiche.
- Passare all'azione: le azioni passate nelle policy di livello amministrativo delegano il processo decisionale ai livelli inferiori. Quando il traffico corrisponde a una regola Pass, la valutazione ignora tutte le restanti regole di livello di amministratore per quel traffico e passa direttamente al livello. NetworkPolicy Ciò consente agli amministratori di delegare esplicitamente il controllo di determinati modelli di traffico ai team applicativi. Ad esempio, potresti utilizzare le regole Pass per delegare la gestione del traffico all'interno dello spazio dei nomi agli amministratori dello spazio dei nomi mantenendo controlli rigorosi sull'accesso esterno.

2. Livello di policy di rete: se nessun criterio di livello di amministrazione corrisponde a Deny o Allow, o se corrisponde un'azione Pass, vengono valutate successivamente le risorse tradizionali e con ambito di namespace. ApplicationNetworkPolicy NetworkPolicy Queste policy forniscono un controllo dettagliato all'interno dei singoli namespace e sono gestite dai team applicativi. Namespace-scoped le policy possono essere solo più restrittive delle policy di amministrazione. Non possono sovrascrivere la decisione di rifiuto di una politica di amministrazione, ma possono limitare ulteriormente il traffico consentito o passato dai criteri di amministrazione.
3. Politiche di amministrazione di livello base: se nessuna politica amministrativa o con ambito di namespace corrisponde al traffico, viene valutato il livello Baseline. ClusterNetworkPolicies Questi forniscono posizioni di sicurezza predefinite che possono essere sostituite da policy con ambito di namespace, consentendo agli amministratori di impostare impostazioni predefinite a livello di organizzazione e offrendo ai team la flessibilità di personalizzarle in base alle esigenze. Le politiche di base vengono valutate in ordine di priorità (prima il numero con la priorità più bassa).
4. Nega predefinita (se nessuna politica corrisponde): questo comportamento di negazione per impostazione predefinita garantisce che siano consentite solo connessioni esplicitamente consentite, mantenendo un elevato livello di sicurezza.

### Applicazione del principio del privilegio minimo

- Inizia con politiche restrittive e aggiungi gradualmente le autorizzazioni in base alle esigenze. Inizia implementando le politiche negate per impostazione predefinita a livello di cluster, quindi aggiungi in modo incrementale le regole di autorizzazione man mano che convalidi i requisiti di connettività legittimi. Questo approccio obbliga i team a giustificare esplicitamente ogni connessione esterna, creando un ambiente più sicuro e verificabile.
- Verifica e rimuovi regolarmente le regole delle policy inutilizzate: le policy di rete possono accumularsi nel tempo man mano che le applicazioni si evolvono, lasciando dietro di sé regole obsolete che espandono inutilmente la superficie di attacco. Implementa un processo di revisione regolare per identificare e rimuovere le regole politiche che non sono più necessarie, assicurandoti che il tuo livello di sicurezza rimanga rigoroso e manutenibile.
- Se possibile, utilizza nomi di dominio specifici anziché schemi generici. Se da un lato i modelli jolly `*.amazonaws.com` offrono praticità, dall'altro consentono l'accesso a un'ampia gamma di servizi. Ove possibile, specifica nomi di dominio esatti, `s3.us-west-2.amazonaws.com` ad esempio per limitare l'accesso solo ai servizi specifici richiesti dalle applicazioni, riducendo il rischio di movimenti laterali in caso di compromissione del carico di lavoro.

### Utilizzo delle policy in EKS DNS-based

- Le regole basate sul DNS definite utilizzando il `ApplicationNetworkPolicy` sono applicabili solo ai carichi di lavoro eseguiti nelle istanze EKS Auto Mode-launched EC2. Se stai eseguendo un cluster in modalità mista (composto da nodi worker EKS Auto e non EKS Auto), le DNS-based regole sono efficaci solo nei nodi di lavoro in modalità EKS Auto (istanze gestite EC2).

### Convalida delle politiche DNS

- Usa cluster di staging che rispecchiano la topologia della rete di produzione per i test: il tuo ambiente di staging dovrebbe replicare l'architettura di rete, le dipendenze esterne e i modelli di connettività della produzione per garantire un test accurato delle policy. Ciò include configurazioni VPC corrispondenti, comportamento di risoluzione DNS e accesso agli stessi servizi esterni richiesti dai carichi di lavoro di produzione.
- Implementa test automatici per percorsi di rete critici: crea test automatici che convalidano la connettività ai servizi esterni essenziali come parte della tua pipeline. CI/CD Questi test dovrebbero verificare che siano consentiti flussi di traffico legittimi mentre le connessioni non autorizzate sono

bloccate, garantendo una convalida continua che le politiche di rete mantengano il corretto livello di sicurezza man mano che l'infrastruttura si evolve.

- Monitora il comportamento delle applicazioni dopo le modifiche alle policy: dopo aver implementato politiche di rete nuove o modificate in produzione, monitora attentamente i log delle applicazioni, i tassi di errore e le metriche delle prestazioni per identificare rapidamente eventuali problemi di connettività. Stabilisci procedure di rollback chiare in modo da poter ripristinare rapidamente le modifiche alle policy se causano un comportamento imprevisto delle applicazioni o interruzioni del servizio.

Interazione con il firewall DNS di Amazon Route 53

Le policy di amministrazione e rete di EKS vengono valutate per prime a livello di pod quando viene avviato il traffico. Se una policy di rete EKS consente l'accesso a un dominio specifico, il pod esegue quindi una query DNS che raggiunge il Route 53 Resolver. A questo punto, vengono valutate le regole del firewall DNS di Route 53. Se il firewall DNS blocca la query sul dominio, la risoluzione DNS fallisce e la connessione non può essere stabilita, anche se la politica di rete EKS lo consente. Ciò crea livelli di sicurezza complementari: le policy di DNS-based rete EKS forniscono un controllo in uscita a livello di pod per i requisiti di accesso specifici delle applicazioni e i limiti di sicurezza multi-tenant, mentre il firewall DNS fornisce protezione contro i domini dannosi noti e applica le blocklist a livello di organizzazione. VPC-wide

## Assegnazione di tag alle sottoreti per la modalità automatica EKS

Se utilizzi la funzionalità di bilanciamento del carico di EKS Auto Mode, devi aggiungere AWS tag alle sottoreti VPC.

### Contesto

Questi tag identificano le sottoreti associate al cluster e, soprattutto, se la sottorete è pubblica o privata.

Le sottoreti pubbliche dispongono di un accesso diretto a Internet tramite un gateway Internet. Sono utilizzate per risorse che devono essere accessibili al pubblico, come i bilanciatori del carico.

Le sottoreti private non dispongono di un accesso diretto a Internet e utilizzano gateway NAT per il traffico in uscita. Vengono utilizzati per risorse interne come i nodi EKS che non necessitano di risorse pubbliche. IPs

Per ulteriori informazioni sui gateway NAT e sui gateway Internet, consulta Connetti il tuo VPC ad altre reti nella Guida per l'utente del cloud privato virtuale (VPC) di Amazon.

## Requisito

Al momento, le sottoreti utilizzate per il bilanciamento del carico tramite la modalità automatica EKS devono avere uno dei seguenti tag.

Sottoreti pubbliche

Le sottoreti pubbliche sono utilizzate per i bilanciatori del carico connesso a Internet. Queste sottoreti devono avere i tag seguenti:

| Chiave | Valore |
| --- | --- |
| `kubernetes.io/role/elb` | `1 o ``` |

Sottoreti private

Le sottoreti private sono utilizzate per i bilanciatori del carico interni. Queste sottoreti devono avere i tag seguenti:

| Chiave | Valore |
| --- | --- |
| `kubernetes.io/role/internal-elb` | `1 o ``` |

## Procedura

Prima di iniziare, identifica quali sottoreti sono pubbliche (con accesso a gateway Internet) e quali sono private (tramite gateway NAT). Avrai bisogno delle autorizzazioni per modificare le risorse VPC.

Console di gestione AWS

1. Apri la console Amazon VPC e accedi alle sottoreti.
2. Seleziona la sottorete da etichettare.
3. Scegli la scheda Tag e seleziona Aggiungi tag.
4. Aggiungi il tag appropriato:

- Per le sottoreti pubbliche: Chiave=`kubernetes.io/role/elb`
- Per sottoreti private: Chiave=`kubernetes.io/role/internal-elb`

5. Imposta Valore su 1 o lascia vuoto.
6. Salva e ripeti per le sottoreti rimanenti.

AWS CLI

Per le sottoreti pubbliche:

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/elb,Value=1
```

Per le sottoreti private:

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/internal-elb,Value=1
```

Sostituisci `subnet-ID` con l'ID della sottorete corrente.

# Generare report di conformità CIS dai nodi Kubernetes utilizzando kubectl debug

Questo argomento descrive come generare report di conformità CIS (Center for Internet Security) per i nodi Amazon EKS utilizzando il comando `kubectl debug`. Il comando ti consente di creare temporaneamente un container di debug su un nodo di Kubernetes ed eseguire controlli di conformità CIS utilizzando lo strumento `apiclient`. Lo strumento `apiclient` fa parte di Bottlerocket OS, il sistema operativo utilizzato dai nodi della modalità automatica di EKS.

## Prerequisiti

Prima di iniziare, assicurati di disporre di:

- Accesso a un cluster Amazon EKS con `kubectl` configurato (la versione deve essere almeno v1.32.0; digitare `kubectl version` per verificare).
- Le autorizzazioni IAM appropriate per il debug dei nodi.
- Un profilo valido che consente operazioni di debug (ad esempio, `sysadmin`).

Per ulteriori informazioni sull'utilizzo dei profili di debug con `kubectl`, consultare Debugging a Pod or Node while applying a profile nella documentazione di Kubernetes.

## Procedura

1. Determina l'ID AWS istanza del nodo su cui vuoi eseguire il rapporto. Utilizzare il seguente comando per elencare tutti i nodi nel cluster. L'ID dell'istanza si trova nella colonna del nome e inizia con `i-`:

```
kubectl get nodes
```

```
NAME                  STATUS   ROLES    AGE   VERSION
i-0ea0ba0f8ef9ad609   Ready    <none>   62s   v1.30.10-eks-1a9dacd
```

2. Esegui il comando seguente, sostituendo `<instance-id>` con l'ID di istanza del nodo su cui si desidera eseguire la query:

```
kubectl debug node/<instance-id> -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023 -- bash -c "yum install -q -y util-linux-core; nsenter -
t 1 -m apiclient report cis --level 1 --format text"
```

   I componenti di questo comando includono:

   - `kubectl debug node/<instance-id>`: crea una sessione di debug sull'ID di istanza EC2 specificato.
   - `-it`: assegna un TTY (shell a riga di comando) e mantiene aperto stdin per un utilizzo interattivo.
   - `--profile=sysadmin`: utilizza il profilo `kubectl` specificato con le autorizzazioni appropriate.
   - `--image=public.ecr.aws/amazonlinux/amazonlinux:2023`: utilizza `amazonlinux:2023` come immagine del container per il debug.
   - `bash -c "…"` : esegue i seguenti comandi in una shell bash:
     - `yum install -q -y util-linux-core`: installa in background il pacchetto di utilità richiesto.
     - `nsenter -t 1 -m`: esegue `nsenter` per inserire il namespace del processo host (PID 1).
     - `apiclient report cis --level 1 --format text`: esegue il report di conformità CIS a livello 1 con output di testo.

3. Esaminare l'output del testo del report.

## Interpretazione dell'output

Il comando genera un report basato su testo che mostra lo stato di conformità di vari controlli CIS. L'output include:

- Controllo CIS individuale IDs
- Descrizione di ciascun controllo
- Lo stato Pass, Fail o Skip per ogni controllo
- Dettagli che spiegano eventuali problemi di conformità

Ecco un esempio di output del report eseguito su un'istanza Bottlerocket:

```
Benchmark name:  CIS Bottlerocket Benchmark
Version:         v1.0.0
Reference:       https://www.cisecurity.org/benchmark/bottlerocket
Benchmark level: 1
Start time:      2025-04-11T01:40:39.055623436Z

[SKIP] 1.2.1     Ensure software update repositories are configured (Manual)
[PASS] 1.3.1     Ensure dm-verity is configured (Automatic)[PASS] 1.4.1     Ensure
 setuid programs do not create core dumps (Automatic)
[PASS] 1.4.2     Ensure address space layout randomization (ASLR) is enabled
 (Automatic)
[PASS] 1.4.3     Ensure unprivileged eBPF is disabled (Automatic)
[PASS] 1.5.1     Ensure SELinux is configured (Automatic)
[SKIP] 1.6       Ensure updates, patches, and additional security software are
 installed (Manual)
[PASS] 2.1.1.1   Ensure chrony is configured (Automatic)
[PASS] 3.2.5     Ensure broadcast ICMP requests are ignored (Automatic)
[PASS] 3.2.6     Ensure bogus ICMP responses are ignored (Automatic)
[PASS] 3.2.7     Ensure TCP SYN Cookies is enabled (Automatic)
[SKIP] 3.4.1.3   Ensure IPv4 outbound and established connections are configured
 (Manual)
[SKIP] 3.4.2.3   Ensure IPv6 outbound and established connections are configured
 (Manual)
[PASS] 4.1.1.1   Ensure journald is configured to write logs to persistent disk
 (Automatic)
[PASS] 4.1.2     Ensure permissions on journal files are configured (Automatic)

Passed:          11
Failed:          0
```

```
Skipped:         4
Total checks:    15
```

Per informazioni sul benchmark, consultare Kubernetes Benchmark dal Center for Internet Security (CIS).

### Risorse correlate

- Bottlerocket CIS Benchmark nella documentazione del sistema operativo Bottlerocket.
- Debug Running Pods nella documentazione di Kubernetes.
- Kubernetes Benchmark dal Center for Internet Security (CIS)

## Abilitare la crittografia dei volumi EBS con chiavi KMS gestite dal cliente per la modalità automatica di EKS

Puoi crittografare il volume root temporaneo per le istanze della modalità automatica di EKS con una chiave KMS gestita dal cliente.

La modalità automatica di Amazon EKS utilizza ruoli collegati ai servizi per delegare le autorizzazioni ad altri AWS servizi durante la gestione di volumi EBS crittografati per i cluster Kubernetes. In questo argomento viene descritto come impostare le policy della chiave necessarie quando si specifica una chiave gestita dal cliente per la crittografia Amazon EBS con la modalità automatica di EKS.

Considerazioni:

- EKS Auto Mode non richiede un'autorizzazione aggiuntiva per utilizzare la chiave AWS gestita predefinita per proteggere i volumi crittografati nel tuo account.
- Questo argomento tratta la crittografia dei volumi effimeri, i volumi root per le istanze. EC2 Per ulteriori informazioni sulla crittografia dei volumi di dati utilizzati per i carichi di lavoro, consulta the section called “Crea StorageClass”.

### Panoramica di

Le seguenti chiavi AWS KMS possono essere utilizzate per la crittografia dei volumi root di Amazon EBS quando EKS Auto Mode avvia le istanze:

- Chiave gestita da AWS : una chiave di crittografia nell'account che Amazon EBS crea, possiede e gestisce. Questa è la chiave di crittografia di default per un nuovo account.

- Chiave gestita dal cliente: una chiave di crittografia personalizzata che l'utente crea, possiede e gestisce.

> **Note**
>
> La chiave deve essere simmetrica. Amazon EBS non supporta le chiavi gestite dal cliente asimmetriche.

## Fase 1: Configurare la policy della chiave

Le chiavi KMS devono avere una policy della chiave che permetta alla modalità automatica di EKS di avviare istanze con volumi Amazon EBS crittografati con una chiave gestita dal cliente.

Configura la policy della chiave con la seguente struttura:

> **Note**
>
> Questa policy include solo le autorizzazioni per la modalità automatica di EKS. La policy della chiave potrebbe richiedere autorizzazioni aggiuntive se altre identità devono utilizzare la chiave o gestire le concessioni.

```
{
    "Version":"2012-10-17",
    "Id": "MyKeyPolicy",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
```

```
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

Assicurati di sostituirlo <account-id> con l'ID dell'account effettivo. AWS

Durante la configurazione della policy della chiave:

- Per eseguire le operazioni di crittografia, il `ClusterServiceRole` deve disporre delle autorizzazioni IAM necessarie per usare la chiave KMS.
- La `kms:GrantIsForAWSResource` condizione garantisce che le sovvenzioni possano essere create solo per i servizi AWS

## Fase 2: Configura NodeClass con la chiave gestita dal cliente

Dopo aver configurato la politica chiave, fai riferimento alla chiave KMS nella configurazione della modalità NodeClass automatica EKS:

```
apiVersion: eks.amazonaws.com/v1
```

```
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Insert existing configuration

  ephemeralStorage:
    size: "80Gi"  # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000    # Range: 3000-16000
    throughput: 125  # Range: 125-1000

    # KMS key for encryption
    kmsKeyID: "arn:aws: kms:<region>:<account-id>:key/<key-id>"
```

Sostituisci i valori di segnaposto con i valori effettivi:

- `<region>`con la tua regione AWS
- `<account-id>`con l'ID AWS del tuo account
- `<key-id>` con ID della chiave KMS

Puoi specificare la chiave KMS utilizzando uno dei seguenti formati:

- ID della chiave KMS: `1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- ARN della chiave KMS: `arn:aws: kms:us-west-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- Nome alias della chiave: `alias/eks-auto-mode-key`
- ARN alias della chiave: `arn:aws: kms:us-west-2:111122223333:alias/eks-auto-mode-key`

Applica la NodeClass configurazione usando kubectl:

```
kubectl apply -f nodeclass.yaml
```

## Risorse correlate

- Creazione di una classe di nodi per Amazon EKS
- Visualizza ulteriori informazioni nella AWS Key Management Service Developer Guide
  - Autorizzazioni per AWS i servizi nelle politiche chiave

- Modificare una policy della chiave
- Sovvenzioni in KMS AWS

## Aggiorna i controlli organizzativi per la modalità automatica di EKS

Alcuni controlli organizzativi possono impedire il corretto funzionamento della modalità automatica di EKS. In tal caso, devi aggiornare questi controlli per consentire alla modalità automatica di EKS di disporre delle autorizzazioni necessarie per gestire le istanze EC2 per conto dell'utente.

modalità automatica di EKS utilizza un ruolo di servizio per l'avvio delle istanze EC2 che supportano i nodi della modalità automatica di EKS. Un ruolo di servizio è un ruolo IAM creato nell'account che un servizio assume per eseguire operazioni nell'account. Le policy di controllo dei servizi (SCP) si applicano sempre alle operazioni eseguite con i ruoli di servizio. Ciò consente a una SCP di inibire le operazioni di Auto Mode. La circostanza più comune si verifica quando una SCP viene utilizzata per limitare le Amazon Machine Image (AMI) che possono essere avviate. Per consentire il funzionamento della modalità automatica di EKS, modifica la SCP per consentire l'avvio di AMI dagli account modalità automatica di EKS.

Puoi anche possibile utilizzare la funzionalità EC2 Allowed AMIs per limitare la visibilità delle AMI in altri account. Se utilizzi questa funzionalità, devi espandere i criteri dell'immagine per includere anche gli account AMI della modalità automatica di EKS nelle regioni di interesse.

### Ad esempio: SCP per bloccare tutte le AMI ad eccezione delle AMI della modalità automatica di EKS

La SCP riportata di seguito impedisce di chiamare `ec2:RunInstances` a meno che l'AMI non appartenga all'account AMI della modalità automatica di EKS per us-west-2 o us-east-1.

> **Note**
>
> È importante non usare la chiave di contesto `ec2:Owner`. Amazon possiede gli account AMI della modalità automatica di EKS e il valore di questa chiave sarà sempre `amazon`. La creazione di un SCP che consenta il lancio di AMI se lo `ec2:Owner` è `amazon` consentirà il lancio di qualsiasi AMI di proprietà di Amazon, non solo quelle per la modalità automatica EKS.

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*::image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "767397842682",
            "992382739861"
          ]
        }
      }
    }
  ]
}
```

## Account AMI della modalità automatica di EKS

AWS gli account che variano in base alla regione ospitano le AMI pubbliche EKS Auto Mode.

| AWS Regione | Account |
|---|---|
| af-south-1 | 471112993317 |
| ap-east-1 | 590183728416 |
| ap-east-2 | 381492200852 |
| ap-northeast-1 | 851725346105 |
| ap-northeast-2 | 992382805010 |
| ap-northeast-3 | 891377407544 |
| ap-south-1 | 975049899075 |
| ap-south-2 | 590183737426 |
| ap-southeast-1 | 339712723301 |

| | |
|---|---|
| ap-southeast-2 | 058264376476 |
| ap-southeast-3 | 471112941769 |
| ap-southeast-4 | 590183863144 |
| ap-southeast-5 | 654654202513 |
| ap-southeast-6 | 905418310314 |
| ap-southeast-7 | 533267217478 |
| ca-central-1 | 992382439851 |
| ca-west-1 | 767397959864 |
| eu-central-1 | 891376953411 |
| eu-central-2 | 381492036002 |
| eu-north-1 | 339712696471 |
| eu-south-1 | 975049955519 |
| eu-south-2 | 471112620929 |
| eu-west-1 | 381492008532 |
| eu-west-2 | 590184142468 |
| eu-west-3 | 891376969258 |
| il-central-1 | 590183797093 |
| me-central-1 | 637423494195 |
| me-south-1 | 905418070398 |
| mx-central-1 | 211125506622 |
| sa-east-1 | 339712709251 |

| | |
|---|---|
| us-east-1 | 992382739861 |
| us-east-2 | 975050179949 |
| us-west-1 | 975050035094 |
| us-west-2 | 767397842682 |
| us-gov-east-1 | 446077414359 |
| us-gov-west-1 | 446098668741 |

### Indirizzo IP pubblico associato

Quando viene chiamato `ec2:RunInstances`, il campo `AssociatePublicIpAddress` per l'avvio di un'istanza viene determinato automaticamente dal tipo di sottorete in cui viene avviata l'istanza. È possibile utilizzare un SCP per far sì che questo valore sia impostato esplicitamente su false, indipendentemente dal tipo di sottorete in cui viene avviato. In questo caso, il NodeClass campo `spec.advancedNetworking.associatePublicIPAddress` può anche essere impostato su false per soddisfare i requisiti dell'SCP.

```
{
      "Sid": "DenyPublicEC2IPAddresses",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
          "BoolIfExists": {
              "ec2:AssociatePublicIpAddress": "true"
          }
      }
  }
```

## Controlla l'implementazione dei carichi di lavoro in Prenotazioni della capacità con la modalità automatica di EKS

Puoi controllare l'implementazione dei carichi di lavoro su Prenotazioni della capacità. La modalità automatica EKS supporta le prenotazioni di On-Demand capacità EC2 (ODCR), i blocchi di capacità EC2 per il machine learning e le prenotazioni di capacità interrompibili (IODCR).

> **Tip**
>
> Per impostazione predefinita, EKS Auto Mode può essere avviato in ODCR aperti tramite open-matching, ma non dà loro priorità. Le istanze avviate tramite open-matching sono etichettate, non. `karpenter.sh/capacity-type: on-demand` `reserved` Per dare priorità all'utilizzo dell'ODCR e etichettare le istanze, configura nella definizione. `karpenter.sh/capacity-type: reserved` `capacityReservationSelectorTerms` NodeClass I Capacity Blocks for ML richiedono sempre `capacityReservationSelectorTerms` e non vengono utilizzati automaticamente.

## Prenotazioni di On-Demand capacità EC2 (ODCR)

Le prenotazioni di On-Demand capacità EC2 (ODCR) ti consentono di riservare la capacità di elaborazione per le tue istanze Amazon EC2 in una zona di disponibilità specifica per qualsiasi durata. Quando si utilizza modalità automatica di EKS, potresti voler controllare se i carichi di lavoro di Kubernetes vengono implementati su queste istanze riservate per massimizzare l'utilizzo della capacità pre-acquistata o per garantire che i carichi di lavoro critici abbiano accesso a risorse garantite.

Per impostazione predefinita, modalità automatica di EKS si avvia automaticamente in ODCR aperti. Tuttavia, configurando `capacityReservationSelectorTerms` su a, puoi controllare in modo esplicito quali ODCR NodeClass vengono utilizzati dai tuoi carichi di lavoro. I nodi forniti utilizzando ODCR configurate avranno `karpenter.sh/capacity-type: reserved` e avranno priorità rispetto a quelli on-demand e spot. Una volta abilitata questa funzionalità, EKS Auto Mode non utilizzerà più automaticamente gli ODCR aperti, ma devono essere selezionati esplicitamente da un, in modo da consentirvi un controllo preciso sull'utilizzo della prenotazione della NodeClass capacità in tutto il cluster.

> **Warning**
>
> Se si esegue la configurazione *`capacityReservationSelectorTerms`* NodeClass in un cluster, EKS Auto Mode non utilizzerà più automaticamente gli ODCR aperti per nessuno dei componenti del cluster. NodeClass

Esempio NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
spec:
  # Optional: Selects upon on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        app: "my-app"
      # Optional owning account ID filter
      owner: "012345678901"
```

Questo esempio NodeClass illustra due approcci per la selezione degli ODCR. Il primo metodo fa riferimento direttamente a una ODCR specifica tramite il relativo ID (`cr-56fac701cc1951b03`). Il secondo metodo utilizza la selezione basata su tag, usando come target le ODCR con il tag `Name: "targeted-odcr"`. Facoltativamente, puoi anche filtrare in base all' AWS account proprietario della prenotazione, il che è particolarmente utile negli scenari tra account o quando lavori con prenotazioni di capacità condivise.

## Etichette di pianificazione per le prenotazioni di capacità

Quando si configura `capacityReservationSelectorTerms` su a NodeClass, i nodi forniti dalle prenotazioni di capacità vengono etichettati con etichette di pianificazione aggiuntive. È possibile utilizzare queste etichette come NodePool requisiti o vincoli di pianificazione dei pod (come l'affinità dei nodi) per controllare il posizionamento del carico di lavoro.

| Etichetta | Esempio | Description |
| --- | --- | --- |
| `eks.amazonaws.com/capacity-reservation-id` | `cr-56fac701cc1951b03` | L'ID della prenotazione di capacità |
| `eks.amazonaws.com/capacity-` | `default` o `capacity-block` | Il tipo di prenotazione della capacità |

| Etichetta | Esempio | Description |
| --- | --- | --- |
| `reservation-type` | | |
| `eks.amazonaws.com/capacity-reservation-interruptible` | `true` o `false` | Se la prenotazione della capacità è interrompibile |

Queste etichette sono presenti solo sui nodi con. `karpenter.sh/capacity-type: reserved`

## Prenotazioni di capacità interrompibili (IODCR)

Le prenotazioni di capacità interrompibile consentono di condividere la capacità di prenotazione della On-Demand capacità non utilizzata con altri carichi di lavoro all'interno dell'organizzazione. Riutilizzando la capacità inutilizzata come ODCR interrompibili, i carichi di lavoro adatti a operazioni flessibili e con tolleranza ai guasti, come l'elaborazione in batch e l'analisi dei dati, possono trarre vantaggio dalla capacità temporaneamente disponibile. I titolari delle prenotazioni possono recuperare la propria capacità in qualsiasi momento, mentre i consumatori di ODCR interrompibili riceveranno un avviso di interruzione prima della chiusura per consentire lo spegnimento o il checkpoint senza intoppi prima della chiusura del nodo.

Le prenotazioni di capacità interrompibile vengono selezionate utilizzando lo stesso metodo utilizzato per un ODCR standard: per ID o per tag. `capacityReservationSelectorTerms` Per controllare se i carichi di lavoro sono pianificati sulla capacità riservata interrompibile o non interrompibile, utilizza l'etichetta di pianificazione. `eks.amazonaws.com/capacity-reservation-interruptible`

Quando la capacità viene recuperata, le istanze in esecuzione ricevono un avviso di interruzione di 2 minuti tramite eventi. EventBridge Dopo il periodo di preavviso, le istanze in esecuzione nella capacità recuperata entrano in uno stato di spegnimento e vengono terminate. La modalità automatica EKS inizierà automaticamente a svuotare i nodi avviati dalle prenotazioni di capacità interrompibile quando viene ricevuto l'avviso di interruzione di 2 minuti, consentendo ai carichi di lavoro di terminare senza problemi prima che il nodo venga ripristinato.

## Blocchi di capacità EC2 per ML

I Capacity Blocks for ML riservano istanze di elaborazione GPU-based accelerate in date future per supportare i carichi di lavoro di machine learning (ML) di breve durata. Le istanze eseguite all'interno di un Capacity Block vengono automaticamente posizionate vicine tra loro all'interno di Amazon UltraClusters EC2, per reti a bassa latenza, su scala petabit e non bloccanti.

Per ulteriori informazioni sulle piattaforme e sui tipi di istanze supportati, consulta Capacity Blocks for ML nella Guida per l'utente di EC2.

Puoi creare una modalità automatica EKS NodeClass che utilizza un Capacity Block per ML, simile a un ODCR (descritto in precedenza).

Le seguenti definizioni di esempio creano tre risorse:

1. A NodeClass che fa riferimento alla tua prenotazione Capacity Block
2. A NodePool che utilizza NodeClass e applica una macchia
3. Una specifica per Pod che tollera il taint e richiede risorse GPU

### Esempio NodeClass

Questo NodeClass fa riferimento a uno specifico Capacity Block for ML tramite il relativo ID di prenotazione. Puoi ottenere questo ID dalla console EC2.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  # Specify your Capacity Block reservation ID
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
```

Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

### Esempio NodePool

Questo NodePool fa riferimento `gpu` NodeClass e specifica una configurazione importante:

- Utilizza solo la capacità riservata impostando `karpenter.sh/capacity-type: reserved`

- Richiede famiglie di istanze GPU specifiche adatte ai carichi di lavoro ML
- Applica un taint `nvidia.com/gpu` per garantire che solo i carichi di lavoro GPU siano pianificati su questi nodi

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: gpu
      requirements:
        - key: eks.amazonaws.com/instance-family
          operator: In
          values:
            - g6
            - p4d
            - p4de
            - p5
            - p5e
            - p5en
            - p6
            - p6-b200
        - key: karpenter.sh/capacity-type
          operator: In
          values:
            - reserved
            # Enable other capacity types
            # - on-demand
            # - spot
      taints:
        - effect: NoSchedule
          key: nvidia.com/gpu
```

Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

### Pod di esempio

Questo pod di esempio dimostra come configurare un carico di lavoro da eseguire sui nodi del blocco di capacità:

- Utilizza un nodeSelector per individuare tipi di GPU specifici (in questo caso, GPU H200)
- Include una tolleranza per la `nvidia.com/gpu` macchia applicata dal NodePool
- Richiede esplicitamente le risorse GPU che utilizzano il tipo di risorsa `nvidia.com/gpu`

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    # Select specific GPU type - uncomment as needed
    # eks.amazonaws.com/instance-gpu-name: l4
    # eks.amazonaws.com/instance-gpu-name: a100
    eks.amazonaws.com/instance-gpu-name: h200
    # eks.amazonaws.com/instance-gpu-name: b200
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
    - "nvidia-smi"
    resources:
      requests:
        # Uncomment if needed
        # memory: "30Gi"
        # cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        # Uncomment if needed
        # memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

Per ulteriori informazioni, consulta Pods nella documentazione Kubernetes.

Risorse correlate

- Capacity Blocks for ML nella Guida per l'utente di Amazon EC2
- Find and purchase Capacity Blocks nella Guida per l'utente di Amazon EC2
- Prenotazioni di capacità interrompibile nella Guida per l'utente di Amazon EC2
- Gestisci le risorse di calcolo per i AI/ML carichi di lavoro su Amazon EKS
- GPU Resource Optimization and Cost Management nella Guida alle best practice di EKS

## Implementazione dei nodi EKS Auto Mode su Local Zones

EKS Auto Mode offre una gestione semplificata dei cluster con il provisioning automatico dei nodi. AWS Local Zones estendono l' AWS infrastruttura alle aree geografiche più vicine agli utenti finali, riducendo la latenza per le applicazioni sensibili alla latenza. Questa guida illustra il processo di implementazione dei nodi EKS Auto Mode su AWS Local Zones, consentendoti di eseguire applicazioni containerizzate con latenza inferiore per gli utenti in aree geografiche specifiche.

Questa guida dimostra anche come utilizzare i limiti e le tolleranze di Kubernetes per garantire che sui nodi della zona locale vengano eseguiti solo carichi di lavoro specifici, aiutandoti a controllare i costi e ottimizzare l'utilizzo delle risorse.

### Prerequisiti

Prima di iniziare a distribuire i nodi EKS Auto Mode su Local Zones, assicurati di avere i seguenti prerequisiti:

- Un cluster EKS Auto Mode esistente
- Hai effettuato l'accesso alla zona locale del tuo account AWS

### Fase 1: Creare una sottorete di zona locale

Il primo passo per implementare i nodi EKS Auto Mode in una zona locale consiste nella creazione di una sottorete in quella zona locale. Questa sottorete fornisce l'infrastruttura di rete per i nodi e consente loro di comunicare con il resto del VPC. Segui le istruzioni Create a Local Zone subnet (nella AWS Local Zones User Guide) per creare una sottorete nella Local Zone prescelta.

> **Tip**
>
> Prendi nota del nome della sottorete della zona locale.

### Fase 2: Creazione NodeClass per la sottorete della zona locale

Dopo aver creato la sottorete Local Zone, è necessario definire una sottorete NodeClass che faccia riferimento a questa sottorete. NodeClass Si tratta di una risorsa personalizzata Kubernetes che specifica gli attributi di infrastruttura per i nodi, incluse le sottoreti, i gruppi di sicurezza e le configurazioni di archiviazione da utilizzare. Nell'esempio seguente, creiamo una NodeClass chiamata «zona locale» che ha come target una sottorete di zona locale in base al suo nome. È inoltre possibile utilizzare l'ID della sottorete. Dovrai adattare questa configurazione per indirizzare la tua sottorete Local Zone.

Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: local-zone
spec:
  subnetSelectorTerms:
    - id: <local-subnet-id>
```

### Fase 3: Crea NodePool con NodeClass e Taint

Una volta NodeClass configurato, ora devi crearne uno NodePool che lo NodeClass utilizzi. A NodePool definisce le caratteristiche di calcolo dei nodi, inclusi i tipi di istanza. NodePool utilizza il NodeClass come riferimento per determinare dove avviare le istanze.

Nell'esempio seguente, creiamo un NodePool riferimento alla nostra «zona locale». NodeClass Aggiungiamo anche una macchia ai nodi per garantire che solo i pod con una tolleranza corrispondente possano essere programmati su questi nodi della Zona Locale. Ciò è particolarmente importante per i nodi della zona locale, che in genere hanno costi più elevati e devono essere utilizzati solo da carichi di lavoro che beneficiano specificamente della latenza ridotta.

Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

```
apiVersion: karpenter.sh/v1
```

```
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        node-type: local-zone
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: local-zone
      taints:
        - key: "aws.amazon.com/local-zone"
          value: "true"
          effect: NoSchedule

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["4", "8", "16", "32"]
```

La contaminazione tra key `aws.amazon.com/local-zone` ed effect `NoSchedule` assicura che i pod senza una tolleranza corrispondente non vengano programmati su questi nodi. Ciò impedisce l'esecuzione accidentale di carichi di lavoro regolari nella zona locale, il che potrebbe comportare costi imprevisti.

## Fase 4: Implementazione dei carichi di lavoro con tolleranza e affinità tra nodi

Per un controllo ottimale sul posizionamento dei carichi di lavoro sui nodi della zona locale, utilizzate entrambi taints/tolerations e l'affinità dei nodi insieme. Questo approccio combinato offre i seguenti vantaggi:

1. Controllo dei costi: The taint garantisce che solo i pod con tolleranze esplicite possano utilizzare risorse potenzialmente costose della Zona Locale.
2. Posizionamento garantito: l'affinità dei nodi garantisce che le applicazioni sensibili alla latenza vengano eseguite esclusivamente nella zona locale, non sui normali nodi del cluster.

Ecco un esempio di distribuzione configurata per l'esecuzione specifica sui nodi della zona locale:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: low-latency-app
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: low-latency-app
  template:
    metadata:
      labels:
        app: low-latency-app
    spec:
      tolerations:
      - key: "aws.amazon.com/local-zone"
        operator: "Equal"
        value: "true"
        effect: "NoSchedule"
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: "node-type"
                operator: "In"
                values: ["local-zone"]
      containers:
      - name: application
        image: my-low-latency-app:latest
        resources:
          limits:
            cpu: "1"
            memory: "1Gi"
          requests:
            cpu: "500m"
            memory: "512Mi"
```

Questa distribuzione ha due configurazioni di pianificazione chiave:

1. La tolleranza consente di programmare i pod sui nodi con la contaminazione. `aws.amazon.com/local-zone`
2. Il requisito di affinità dei nodi garantisce che questi pod funzionino solo sui nodi con l'etichetta. `node-type: local-zone`

Insieme, garantiscono che l'applicazione sensibile alla latenza venga eseguita solo sui nodi della zona locale e che le applicazioni normali non consumino le risorse della zona locale a meno che non siano configurate esplicitamente per farlo.

### Passaggio 5: verifica con la console AWS

Dopo aver configurato le NodeClass distribuzioni e le distribuzioni, è necessario verificare che i nodi vengano forniti nella zona locale come previsto e che i carichi di lavoro vengano eseguiti su di essi. NodePool È possibile utilizzare la console di AWS gestione per verificare che le EC2 istanze vengano avviate nella sottorete della zona locale corretta.

Inoltre, puoi controllare l'elenco dei nodi Kubernetes utilizzando `kubectl get nodes -o wide` per confermare che i nodi si uniscano al cluster con le etichette e le sfumature corrette:

```
kubectl get nodes -o wide
kubectl describe node <node-name> | grep -A 5 Taints
```

Puoi anche verificare che i pod del carico di lavoro siano pianificati sui nodi della zona locale:

```
kubectl get pods -o wide
```

Questo approccio garantisce che su questi nodi vengano pianificati solo i carichi di lavoro che tollerano specificamente la contaminazione della zona locale, aiutandoti a controllare i costi e a utilizzare nel modo più efficiente le risorse della zona locale.

## Configurare le impostazioni di sicurezza avanzate per i nodi

Questo argomento descrive come configurare le impostazioni di sicurezza avanzate per i nodi Amazon EKS Auto Mode utilizzando le `advancedSecurity` specifiche nella tua classe di nodi.

### Prerequisiti

Prima di iniziare, assicurati di disporre dei seguenti elementi:

- Un cluster della modalità automatica di Amazon EKS. Per ulteriori informazioni, consulta the section called "Creazione di un cluster".
- `kubectl` installato e configurato. Per ulteriori informazioni, consulta Configurazione.
- Comprensione della configurazione della classe di nodi. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

## Configura le impostazioni di sicurezza avanzate

Per configurare le impostazioni di sicurezza avanzate per i tuoi nodi, imposta i `advancedSecurity` campi nella specifica della tua classe di nodi:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: security-hardened
spec:
  role: MyNodeRole

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  advancedSecurity:
    # Enable FIPS-compliant AMIs (US regions only)
    fips: true

    # Configure kernel lockdown mode
    kernelLockdown: "integrity"
```

Applica questa configurazione:

```
kubectl apply -f nodeclass.yaml
```

Fai riferimento a questa classe di nodi nella configurazione del tuo pool di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

### Descrizioni dei campi

- `fips`(booleano, opzionale): se impostato su`true`, effettua il provisioning dei nodi utilizzando moduli crittografici AMIs convalidati FIPS 140-2. Questa impostazione seleziona la conformità a FIPS; i clienti sono responsabili della gestione dei propri requisiti di conformità AMIs. Per ulteriori informazioni, consulta Conformità FIPS.AWS Default: `false`.
- `kernelLockdown`(stringa, opzionale): Controlla la modalità del modulo di sicurezza Kernel Lockdown. Valori accettati:
  - `integrity`: blocca i metodi per sovrascrivere la memoria del kernel o modificare il codice del kernel. Impedisce il caricamento dei moduli del kernel non firmati.
  - `none`: disattiva la protezione dal blocco del kernel.

    Per ulteriori informazioni, consultate la documentazione relativa al blocco del kernel Linux.

### Considerazioni

- Gli standard FIPS AMIs sono disponibili nelle regioni Stati Uniti orientali/occidentali, ( AWS Stati Uniti) e Canada ( AWS GovCloud Centro/Ovest). AWS Per ulteriori informazioni,AWS consulta Conformità FIPS.
- Durante l'utilizzo`kernelLockdown: "integrity"`, assicurati che i tuoi carichi di lavoro non richiedano il caricamento di moduli del kernel non firmati o la modifica della memoria del kernel.

### Risorse correlate

- the section called “Creazione di una classe di nodi”- Guida completa alla configurazione della classe di nodi
- the section called “Creazione di pool di nodi”- Configurazione del pool di nodi

## Ottimizzazione dei costi in modalità automatica EKS

EKS Auto Mode ottimizza continuamente i costi di elaborazione del cluster attraverso il consolidamento, il bin-packing e il corretto dimensionamento. Tuttavia, alcune configurazioni del carico di lavoro possono impedire queste ottimizzazioni. Questo argomento spiega come funziona l'ottimizzazione dei costi, cosa può bloccarla e come configurare il cluster per mantenere l'efficienza dei costi.

## In che modo EKS Auto Mode ottimizza i costi

EKS Auto Mode riduce i costi di elaborazione attraverso i seguenti meccanismi:

- Bin-packing— Quando si pianificano i pod sui nodi, EKS Auto Mode seleziona i tipi di istanza che corrispondono strettamente alle richieste di risorse aggregate, riducendo al minimo la capacità inutilizzata.
- Consolidamento: EKS Auto Mode valuta periodicamente i nodi in esecuzione e li sostituisce o li rimuove quando i carichi di lavoro possono essere eseguiti su un numero inferiore o meno costoso di istanze.
- Right-sizing— Man mano che i carichi di lavoro si riducono, EKS Auto Mode consolida i pod su nodi più piccoli e termina le istanze sottoutilizzate.

Queste ottimizzazioni vengono eseguite continuamente senza intervento manuale. Tuttavia, alcune annotazioni e NodePool configurazioni dei pod possono impedire che il consolidamento abbia effetto.

## Built-in pool di nodi e barriere di costo

I pool integrati `general-purpose` e quelli a `system` nodi applicano già diverse impostazioni predefinite di protezione dei costi:

- Famiglie di istanze limitate a C, M e R: non sono consentiti tipi di istanze accelerati (P, G, Inf, Trn) o esotici.
- On-demand solo capacità: istanze No Spot, che evitano il tasso di abbandono dovuto alle interruzioni ma comportano anche l'assenza di risparmi Spot.
- Generazione 5 o successiva: sono escluse le generazioni di istanze più vecchie e meno efficienti in termini di costi.

Se utilizzi solo i pool di nodi integrati, trarrai già vantaggio da questi guardrail. Le indicazioni contenute in questo argomento sull'esclusione delle famiglie di istanze e sulla limitazione delle dimensioni delle istanze sono particolarmente importanti quando si creano istanze personalizzate NodePools, che non ereditano queste restrizioni.

Tuttavia, anche con i pool di nodi integrati, le seguenti sezioni si applicano comunque all'utente:

- the section called "Cosa blocca il consolidamento"— L'`do-not-disrupt`annotazione e i PDB restrittivi bloccano il consolidamento indipendentemente da chi ha fornito il nodo. NodePool

- the section called "Utilizzate NodePool i limiti come massimale di costo"— I pool di nodi integrati non dispongono di risorse configurate. `limits` Se i carichi di lavoro sono scalabili in modo significativo, prendi in considerazione la possibilità di crearne uno personalizzato NodePool con limiti anziché affidarti ai pool integrati illimitati.
- the section called "Ciclo di vita e costo dei nodi"— La sovrapposizione di sostituzione dei nodi si applica a tutti i nodi, compresi quelli forniti dai pool integrati.

| Guardrail | Built-in pool di nodi | Personalizzato NodePools |
|---|---|---|
| Esclusione accelerata delle istanze | Applicato | È necessario configurare |
| Limiti di dimensione delle istanze | Non impostato | È necessario configurare |
| Risorsa `limits` (CPU/memory soffitto) | Non impostato | È necessario configurare |
| On-demand solo | Applicato | Scegli tu (Spot/On-Demand) |
| Protezione dal consolidamento (`do-not-disrupt` /PDB) | La tua responsabilità | La tua responsabilità |

## Cosa blocca il consolidamento

Il consolidamento viene bloccato quando EKS Auto Mode determina che l'interruzione di un nodo violerebbe i requisiti di disponibilità di un carico di lavoro. Le seguenti configurazioni impediscono il consolidamento:

### L'annotazione «do-not-disrupt»

L'`karpenter.sh/do-not-disrupt`annotazione indica a EKS Auto Mode di conservare un nodo finché il pod annotato è in esecuzione su di esso. Ciò impedisce che il nodo venga consolidato, sostituito o terminato, anche se il nodo è sottoutilizzato.

```
metadata:
```

```
  annotations:
    karpenter.sh/do-not-disrupt: "true"
```

**Important**

Implicazione dei costi: quando un pod contiene l'`do-not-disrupt`annotazione, il nodo su cui viene eseguita è esente dal consolidamento. Ciò significa che:

- Il nodo continua a funzionare alla dimensione corrente dell'istanza indipendentemente dall'utilizzo effettivo.
- L'utilizzo di vCPU e memoria su quel nodo può rimanere elevato anche se la domanda del carico di lavoro diminuisce.
- Se più pod su più nodi contengono questa annotazione, il consolidamento a livello di cluster viene notevolmente ridotto, con conseguente aumento dei costi sostenuti.

L'`do-not-disrupt`annotazione è un meccanismo di disponibilità. Non tiene conto dei costi. Usalo solo per carichi di lavoro in cui un'interruzione durante l'esecuzione causa la perdita di dati o una significativa rilavorazione, ad esempio processi in batch di lunga durata o processi stateful senza checkpoint.

Alternative da considerare:

- Pod Disruption Budgets (PDB): utilizzate i PDB per controllare la frequenza delle interruzioni anziché bloccarla completamente. I PDB consentono di procedere al consolidamento garantendo al contempo la disponibilità di un numero minimo di repliche.
- Shorter-lived carichi di lavoro: per CI/CD i runner e gli agenti di compilazione, consentite le interruzioni e affidatevi alla logica di riprova integrata nel sistema CI anziché utilizzarla. `do-not-disrupt`
- Time-limited annotazioni: applicale `do-not-disrupt` solo per la durata di un'operazione critica, quindi rimuovile a livello di codice al termine dell'operazione.

Pod Disruption Budget (PDB)

I PDB con un numero di repliche `minAvailable` pari `maxUnavailable: 0` o uguale a quello corrente bloccano di fatto ogni consolidamento per i pod interessati. Esamina i tuoi PDB per assicurarti che consentano l'interruzione di almeno un pod alla volta.

## Utilizzate NodePool i limiti come massimale di costo

NodePool `limits`stabilisci un limite massimo alle risorse di elaborazione totali che una macchina NodePool può fornire. Quando viene raggiunto il limite, EKS Auto Mode interrompe il lancio di nuovi nodi a tale scopo. NodePool Ciò accade anche se i pod sono in sospeso.

`limits`Utilizzatelo come barriera ai costi, in particolare per NodePools carichi di lavoro non di produzione, di test o a intermittenza in cui la scalabilità illimitata non è appropriata.

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: ci-runners
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m"]
  limits:
    cpu: "500"
    memory: 1000Gi
```

In questo esempio, `ci-runners` NodePool non può superare 500 vCPU o 1000 GiB di memoria in totale su tutti i nodi di cui fornisce il provisioning. I pod che superano questo limite rimangono invariati fino a quando la capacità non viene `Pending` liberata.

> **Tip**
>
> Imposta `limits` in base alla dimensione massima prevista del burst più un buffer per la sostituzione dei nodi. Controlla regolarmente il tuo NodePool utilizzo e regola i limiti man mano che cambiano i modelli di carico di lavoro.

## Escludi le famiglie di istanze per il controllo dei costi

Per impostazione predefinita, EKS Auto Mode seleziona tra un'ampia gamma di tipi di istanze per massimizzare la flessibilità di pianificazione. Per i carichi di lavoro che non richiedono hardware specializzato, limita le famiglie di istanze per evitare il lancio di costosi tipi di istanze.

### Escludi le istanze accelerate

Se i tuoi carichi di lavoro non richiedono risorse GPU o acceleratori, escludi le famiglie di istanze accelerate dalle tue. NodePool In questo modo si evitano scenari in cui le istanze accelerate vengono selezionate durante i vincoli di capacità.

```
spec:
  template:
    spec:
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
```

Specificando solo le categorie ottimizzate per il calcolo, per uso generico e ottimizzate per la memoria, si escludono dalla selezione le famiglie di istanze accelerate (P, G, Inf, Trn) e altre famiglie di istanze specializzate.

### In che modo la selezione delle istanze interagisce con i vincoli di capacità

EKS Auto Mode riduce la priorità ai tipi di istanza accelerati ed esotici durante la normale selezione delle istanze. Tuttavia, quando si verificano errori di avvio prolungati, EKS Auto Mode verrà avviata da qualsiasi tipo di istanza disponibile rimanente per dare priorità alla disponibilità del carico di lavoro. Ad esempio, ciò accade quando le quote di servizio EC2 sono temporaneamente esaurite per tutti i tipi di istanze preferiti.

Per evitare questo comportamento fallback, limita esplicitamente i tuoi NodePool requisiti alle sole categorie di istanze richieste dal tuo carico di lavoro. Quando i tipi preferiti non sono disponibili e non sono consentiti altri tipi dalla NodePool configurazione, i pod rimangono invariati anziché essere programmati su `Pending` istanze costose.

### Limita le dimensioni delle istanze

Oltre a limitare le famiglie di istanze, puoi limitare la dimensione massima delle istanze all'interno delle tue. NodePool La limitazione delle dimensioni delle istanze limita l'esposizione ai costi di

ogni singolo nodo che non può essere consolidato. Ad esempio, un nodo bloccato da un'`do-not-disrupt`annotazione non può ridursi anche se il carico di lavoro è ridotto.

Utilizzate l'`eks.amazonaws.com/instance-cpu`etichetta per limitare le dimensioni massime delle istanze in base ai vostri requisiti: NodePool

```
requirements:
  - key: "eks.amazonaws.com/instance-cpu"
    operator: Lte
    values: ["32"]
```

Questa configurazione impedisce a EKS Auto Mode di avviare istanze più grandi di 32 vCPU al suo interno. NodePool

Per identificare le opportunità di ottimizzazione in un cluster esistente, esamina le istanze più grandi in esecuzione. Se il consolidamento dei nodi di grandi dimensioni viene costantemente bloccato, il costo per nodo di tale capacità inattiva è proporzionalmente più elevato.

## Schemi consigliati per carichi di lavoro intensi

CI/CD le pipeline, i processi in batch e i runner temporanei creano modelli di burst-and-idle che richiedono una configurazione specifica per mantenere l'efficienza dei costi.

Impostazioni predefinite consigliate per carichi di lavoro intensi

| Configurazione | Raccomandazione |
| --- | --- |
| `do-not-disrupt` | Non utilizzare per i corridori. CI/CD Affidati invece ai meccanismi di riprova e coda del tuo sistema CI. |
| NodePool `limits` | Imposta un CPU/memory limite in base alla concorrenza massima prevista più il buffer per la sostituzione dei nodi e la sovrapposizione. |
| Categorie di istanze | Limitato a `c` e `m` famiglie. Escludi le famiglie di istanze accelerate (P, G, Inf, Trn) per carichi di lavoro non GPU. |
| Dimensioni dell'istanza | Prendi in considerazione la possibilità di limitarsi a dimensioni moderate (ad esempio, 4-32 vCPU) per |

| Configurazione | Raccomandazione |
| --- | --- |
| | limitare l'esposizione ai costi di ogni singolo nodo che viene bloccato dal consolidamento. |
| Tempi di consolidamento | Usa l'impostazione predefinita `consolidateAfter` . Evita di impostare ritardi prolungati che mantengano online la capacità massima dopo che i corridori hanno completato il gioco. |
| Tipo capacità | Usa le istanze Spot per i corridori con tolleranza ai guasti. Combinalo con On-Demand for build agent che mantengono lo stato durante l'esecuzione. |

Esempio: CI runner NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: ci-runners
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: Lte
          values: ["32"]
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["spot", "on-demand"]
  limits:
    cpu: "500"
    memory: 1000Gi
  disruption:
```

```
    consolidationPolicy: WhenEmptyOrUnderutilized
    consolidateAfter: 30s
```

Questa configurazione:

- Limita a famiglie di istanze convenienti
- Limita la NodePool capacità totale a 500 vCPU
- Consente un consolidamento aggressivo (30 secondi dopo la rimozione dei pod)
- Consente sia Spot che Capacity On-Demand

## Ciclo di vita e costo dei nodi

La modalità automatica EKS sostituisce i nodi con discontinuità quando si discostano dalle specifiche desiderate (ad esempio, dopo il rilascio di una nuova AMI in modalità automatica) o quando la durata del nodo si avvicina alla scadenza. Durante la graziosa sostituzione:

- Un nuovo nodo sostitutivo viene lanciato e diventa pronto.
- I pod vengono scaricati dal vecchio nodo, rispettando i Pod Disruption Budgets.
- Per un breve periodo, sia il vecchio nodo che quello sostitutivo funzionano contemporaneamente.

Per i cluster con nodi grandi o numerosi, questa sovrapposizione può comportare aumenti periodici dei costi. Per ridurre al minimo l'impatto:

- Esamina i budget per le interruzioni: assicurati che i budget previsti per le interruzioni consentano un esaurimento tempestivo. I budget restrittivi prolungano il periodo di sovrapposizione durante il quale sono operativi sia i vecchi che i nuovi nodi.
- Right-size istanze: le istanze più piccole riducono il costo assoluto della finestra di sovrapposizione.
- Riduzione della durata massima dei nodi: valori di scadenza più bassi (ad esempio, 7 giorni) creano eventi di sostituzione più frequenti ma di dimensioni inferiori. Ciò ripartisce il costo in modo più uniforme nel tempo anziché concentrarlo.

Per ulteriori informazioni sul ciclo di vita dei nodi, consulta Informazioni sulle istanze gestite in modalità automatica di Amazon EKS.

# Scopri come funziona EKS Auto Mode

Utilizza questo capitolo per scoprire come funzionano i componenti dei cluster di Amazon EKS Auto Mode.

Argomenti


- Informazioni sulle istanze gestite della modalità automatica di Amazon EKS
- Informazioni su identità e accesso in modalità automatica di EKS
- Informazioni sulla rete VPC e sul bilanciamento del carico in modalità automatica di EKS


## Informazioni sulle istanze gestite della modalità automatica di Amazon EKS

Questo argomento spiega come la modalità automatica di Amazon EKS gestisce le istanze Amazon EC2 nel cluster EKS. Quando abiliti la modalità automatica EKS, le risorse di elaborazione del cluster vengono fornite e gestite automaticamente da EKS, cambiando la modalità in cui interagisci con le istanze EC2 che fungono da nodi nel cluster.

Comprendere come la modalità automatica di Amazon EKS gestisce le istanze è essenziale per pianificare la strategia di implementazione dei carichi di lavoro e le procedure operative. A differenza delle istanze EC2 tradizionali o dei gruppi di nodi gestiti, queste istanze seguono un modello di ciclo di vita diverso in cui EKS si assume la responsabilità di molti aspetti operativi, limitando al contempo determinati tipi di accesso e personalizzazione.

La modalità automatica di Amazon EKS automatizza le attività di routine per la creazione di nuove istanze EC2 e le collega come nodi al cluster EKS. La modalità automatica EKS rileva quando un carico di lavoro non può adattarsi ai nodi esistenti e crea una nuova istanza EC2.

La modalità automatica di Amazon EKS è responsabile di creazione, eliminazione e applicazione di patch per le istanze EC2. Sei responsabile dei container e dei pod implementati sull'istanza.

Le istanze EC2 create dalla modalità automatica EKS sono diverse dalle altre istanze EC2, in quanto sono gestite. Queste istanze gestite sono di proprietà di EKS e sono più limitate. Non puoi accedere o installare direttamente il software sulle istanze gestite dalla modalità automatica EKS.

AWS suggerisce di eseguire EKS Auto Mode o Karpenter autogestito. Puoi installarli sia durante una migrazione che in una configurazione avanzata. Se hai già installato entrambi, configura i pool di nodi in modo tale che i carichi di lavoro siano associati a Karpenter o modalità automatica EKS.

Per ulteriori informazioni, consulta Amazon EC2 managed instances nella Guida per l'utente di Amazon EC2.

## Tabella di confronto

| Istanza EC2 standard | Istanza gestita dalla modalità automatica EKS |
|---|---|
| Sei responsabile dell'applicazione di patch e dell'aggiornamento dell'istanza. | AWS corregge e aggiorna automaticamente l'istanza. |
| EKS non è responsabile del software sull'istanza. | EKS è responsabile di determinati software sull'istanza, come `kubelet`, il runtime del container e il sistema operativo. |
| Puoi eliminare l'istanza EC2 utilizzando l'API EC2. | EKS determina il numero di istanze implementate nell'account. Se elimini un carico di lavoro, EKS ridurrà il numero di istanze nell'account. |
| Puoi usare SSH per accedere all'istanza EC2. | Puoi implementare pod e container nell'istanza gestita. |
| Puoi determinare il sistema operativo e l'immagine (AMI). | AWS determina il sistema operativo e l'immagine. |
| Puoi implementare carichi di lavoro che si basano sulle funzionalità di Windows o Ubuntu. | Puoi distribuire container basati su Linux, ma senza dipendenze specifiche del sistema operativo. |
| Puoi determinare il tipo e la famiglia di istanze da avviare. | AWS determina il tipo e la famiglia di istanze da avviare. Puoi utilizzare un pool di nodi per limitare i tipi di istanze tra cui la modalità automatica EKS effettua la selezione. |

La seguente funzionalità è efficace sia per le istanze gestite che per le istanze EC2 standard:

- Puoi utilizzare lo storage delle istanze come storage temporaneo per i carichi di lavoro.

### Supporto AMI

Con la modalità automatica EKS, AWS determina l'immagine (AMI) utilizzata per i nodi di calcolo. AWS monitora il lancio delle nuove versioni AMI EKS Auto Mode. Se riscontri problemi di carico di lavoro relativi a una versione AMI, crea una richiesta di supporto. Per ulteriori informazioni, vedere Creazione di casi di supporto e gestione dei casi nella Guida AWS per l'utente di supporto.

In genere, EKS rilascia una nuova AMI ogni settimana contenente CVE e correzioni di sicurezza.

## Visibilità gestita delle risorse

> **Note**
>
> A partire dal 22 aprile 2026, le nuove istanze gestite di Amazon EC2 e le risorse associate (ad esempio, modelli di lancio EC2, volumi EBS e interfacce di rete (ENI)) create da EKS Auto Mode sono nascoste dalle visualizzazioni della console EC2 e dalle operazioni degli elenchi di API per impostazione predefinita. `describe` Le risorse gestite già presenti nel tuo account prima di tale data rimangono visibili. Puoi modificare questo comportamento utilizzando le impostazioni di visibilità delle risorse gestite.

Amazon EC2 fornisce impostazioni di visibilità delle risorse gestite che controllano se le risorse gestite vengono visualizzate nelle visualizzazioni della console EC2 e nelle operazioni degli elenchi di API, ad esempio. `DescribeInstances`

Quando le risorse gestite sono nascoste, le istanze gestite EKS Auto Mode e le risorse associate (volumi EBS, modelli di avvio e interfacce di rete) non vengono visualizzate nella console EC2 o nelle risposte API. `describe` Ciò può semplificare le dashboard di governance, ridurre il rumore negli strumenti di osservabilità e prevenire falsi positivi negli scanner CSPM (Cloud Security Posture Management) che segnalano le risorse gestite come configurazioni errate dei clienti.

Le impostazioni di visibilità si applicano a livello di account a tutti i principali IAM. Non è possibile mostrare o nascondere selettivamente le risorse gestite in base al tipo di risorsa o al servizio che le ha create. Le risorse nascoste rimangono completamente operative e fatturabili.

Puoi modificare le impostazioni di visibilità in qualsiasi momento tramite la console Amazon EC2 o l' AWS interfaccia a riga di comando. Per ulteriori informazioni, consulta le impostazioni di visibilità delle risorse gestite nella Amazon EC2 User Guide.

> **Note**
>
> Anche quando le risorse gestite sono nascoste dalle visualizzazioni della console EC2 e dalle API degli elenchi, puoi comunque visualizzare le istanze EKS Auto Mode tramite:
>
> - La console Amazon EKS, nella scheda Compute del cluster.
> - API Kubernetes (ad esempio). `kubectl get nodes`
> - Interrogazioni dirette all'API EC2 per ID di istanza (ad esempio,). `describe-instances --instance-ids i-0123456789abcdef0`
> - L'`DescribeInstances`API con il parametro. `include-managed-resources`
> - La console EC2, dopo aver modificato l'impostazione di visibilità delle risorse gestite.

## Riferimento all'istanza supportata dalla modalità automatica EKS

La modalità automatica EKS crea solo istanze dei tipi supportati e che soddisfano un requisito di dimensione minima.

modalità automatica di EKS supporta i tipi di istanza indicati di seguito:

| Family | Tipi di istanza |
|---|---|
| Compute Optimized (C) | c9g, c9gd, c8a, c8g, c8gb, c8gd, c8gn, c8i, c8i-flex, c8ib, c8id, c7a, c7g, c7gd, c7gn, c7i, c7i-flex, c6a, c6g, c6gd, c6gn, c6i, c6id, c6in, c5, c5a, c5ad, c5d, c5n, c4 |
| General Purpose (M) | m9gd, m8a, m8azn, m8g, m8gb, m8gd, m8gn, m8i, m8i-flex, m8id, m7a, m7g, m7gd, m7i, m7i-flex, m6a, m6g, m6gd, m6i, m6id, m6idn, m6idn, m6idn, m6dn 6 pollici, m5, m5a, m5ad, m5d, m5dn, m5n, m5n, m5zn, m4 |
| Memoria ottimizzata (R) | r9g, r8a, r8g, r8gb, r8gd, r8gn, r8i, r8i-flex, r8id, r7d, r7g, r7gd, r7i, r7iz, r6a, r6g, r6gd, r6d, r6id, r6id, r6idn, r6in, r5, r5r a, r5ad, r5b, r5d, r5dn, r5n, r4 |
| Burstable (T) | t4g, t3, t3a, t2 |
| Memoria elevata (Z/X) | z1d, x8aedz, x8g, x8i, x2gd |

| Family | Tipi di istanza |
| --- | --- |
| Archiviazione I/D ottimizzata () | i8g, i8ge, i7i, i7ie, i4g, is4gen, im4gn, i4i, i3, i3en, d3, d3en |
| Trn/P/Inf/ GCalcolo accelerato () | trn2, trn1, trn1n, p6-b200, p6-b300, p5, p5e, p5en, p4d, p4de, p3, p3dn, inf2, inf1, g7e, g6, gr6, g6e, g5, g5g, g4ad, g4dn |
| Calcolo ad alte prestazioni () X2/ HPC | x2idn, x2iedn, x2iezn, hpc8a |

Inoltre, la modalità automatica EKS creerà solo istanze EC2 che soddisfano i seguenti requisiti:

- Più di 1 CPU
- Le dimensioni delle istanze non sono nano, micro o piccole

Per ulteriori informazioni, consulta Amazon EC2 instance type naming conventions.

## Servizio di metadati dell'istanza

- La modalità automatica EKS applica IMDSv2 con un limite di hop di 1 per impostazione predefinita, aderendo alle migliori pratiche di sicurezza. AWS
- Questa configurazione predefinita non può essere modificata in modalità automatica.
- Per i componenti aggiuntivi che in genere richiedono l'accesso all'IMDS, fornisci i parametri (come la AWS regione) durante l'installazione per evitare ricerche IMDS. Per ulteriori informazioni, consulta the section called "Campi che puoi personalizzare".
- Se un Pod richiede assolutamente l'accesso IMDS quando viene eseguito in modalità automatica, il Pod deve essere configurato per funzionare con `hostNetwork: true`. Ciò consente al Pod di accedere direttamente al servizio di metadati dell'istanza.
- Considerare le implicazioni sulla sicurezza quando si concede ai Pod l'accesso ai metadati dell'istanza.

Per ulteriori informazioni sul servizio di metadati di istanza (IMDS) di Amazon EC2, consulta Configure the Instance Metadata Service options nella Guida per l'utente di Amazon EC2.

### Considerazioni

- Se lo storage temporaneo configurato in NodeClass è inferiore allo storage locale NVMe per l'istanza, EKS Auto Mode elimina la necessità della configurazione manuale intraprendendo automaticamente le seguenti azioni:
  - Utilizza un volume inferiore di dati Amazon EBS (20 GiB) per ridurre i costi.
  - Formatta e configura l'archiviazione locale NVMe per l'uso temporaneo dei dati. Ciò include la configurazione di un array RAID 0 se sono presenti più unità NVMe.
- Quando `ephemeralStorage.size` è uguale o superiore alla capacità NVMe locale, si verificano le seguenti operazioni:
  - La modalità automatica salta il volume EBS piccolo.
  - Le unità NVMe sono esposte direttamente al carico di lavoro.
- Amazon EKS Auto Mode non supporta le seguenti azioni del servizio AWS Fault Injection:
  - `ec2:RebootInstances`
  - `ec2:SendSpotInstanceInterruptions`
  - `ec2:StartInstances`
  - `ec2:StopInstances`
  - `ec2:TerminateInstances`
  - `ec2:PauseVolumeIO`
- Amazon EKS Auto Mode supporta le azioni AWS Fault Injection Service EKS Pod. Per ulteriori informazioni, consulta Managing Fault Injection Service experiments e Use the AWS FIS aws:eks:pod actions nella Guida per l' AWS utente di Resilience Hub.
- Non devi installare il `Neuron Device Plugin` sui nodi della modalità automatica EKS.

  Se hai altri tipi di nodi nel cluster, devi configurare il plug-in Neuron Device in modo tale che non venga eseguito su nodi in modalità automatica. Per ulteriori informazioni, consulta the section called "Controllare l'implementazione".

## Informazioni su identità e accesso in modalità automatica di EKS

Questo argomento descrive i ruoli e le autorizzazioni di Identity and Access Management (IAM) necessari per utilizzare modalità automatica di EKS. La modalità automatica di EKS utilizza due ruoli IAM primari: un ruolo IAM del cluster e uno del nodo. Questi ruoli lavorano in combinazione con EKS

Pod Identity e le voci di accesso EKS per fornire una gestione completa degli accessi per i cluster EKS.

Quando configuri EKS Auto Mode, dovrai configurare questi ruoli IAM con autorizzazioni specifiche che consentano ai AWS servizi di interagire con le risorse del tuo cluster. Ciò include le autorizzazioni per la gestione delle risorse di elaborazione, dei volumi di archiviazione, dei bilanciatori del carico e dei componenti di rete. La comprensione di queste configurazioni dei ruoli è essenziale per il corretto funzionamento e la sicurezza del cluster.

In EKS Auto Mode, i ruoli AWS IAM vengono automaticamente mappati alle autorizzazioni Kubernetes tramite le voci di accesso EKS, eliminando la necessità di configurazioni manuali o associazioni personalizzate. `aws-auth` ConfigMaps Quando si crea un nuovo cluster in modalità automatica, EKS crea automaticamente le autorizzazioni Kubernetes corrispondenti utilizzando le voci di Access, assicurando che i AWS servizi e i componenti del cluster abbiano i livelli di accesso appropriati sia all'interno del sistema di autorizzazione che a quello di Kubernetes. AWS Questa integrazione automatizzata riduce la complessità della configurazione e aiuta a prevenire i problemi relativi alle autorizzazioni che si verificano comunemente durante la gestione dei cluster EKS.

## Ruolo IAM del cluster

Il ruolo Cluster IAM è un ruolo di AWS Identity and Access Management (IAM) utilizzato da Amazon EKS per gestire le autorizzazioni per i cluster Kubernetes. Questo ruolo concede ad Amazon EKS le autorizzazioni necessarie per interagire con altri AWS servizi per conto del cluster e viene automaticamente configurato con le autorizzazioni Kubernetes utilizzando le voci di accesso EKS.

- È necessario AWS allegare politiche IAM a questo ruolo.
- La modalità automatica di EKS assegna automaticamente le autorizzazioni Kubernetes a questo ruolo utilizzando le voci di accesso EKS.
- Con EKS Auto Mode, AWS suggerisce di creare un singolo ruolo IAM del cluster per AWS account.
- AWS suggerisce di assegnare un nome a questo ruolo`AmazonEKSAutoClusterRole`.
- Questo ruolo richiede autorizzazioni per più AWS servizi per gestire le risorse, tra cui volumi EBS, Elastic Load Balancer e istanze EC2.
- La configurazione suggerita per questo ruolo include più policy IAM AWS gestite, relative alle diverse funzionalità di EKS Auto Mode.
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicyV2`
  - `AmazonEKSLoadBalancingPolicy`

- `AmazonEKSNetworkingPolicy`
- `AmazonEKSClusterPolicy`

Per ulteriori informazioni sul ruolo IAM del cluster e sulle politiche IAM AWS gestite, vedere:

- the section called " AWS politiche gestite"
- the section called "Ruolo IAM del cluster"

Per ulteriori informazioni sull'accesso a Kubernetes, consulta:

- the section called "Rivedere le policy di accesso"

## Ruolo IAM del nodo

Il ruolo Node IAM è un ruolo di AWS Identity and Access Management (IAM) utilizzato da Amazon EKS per gestire le autorizzazioni per i nodi di lavoro nei cluster Kubernetes. Questo ruolo concede alle istanze EC2 in esecuzione come nodi Kubernetes le autorizzazioni necessarie per interagire con AWS servizi e risorse e viene configurato automaticamente con le autorizzazioni RBAC di Kubernetes utilizzando le voci di accesso EKS.

- È necessario allegare politiche IAM a questo ruolo. AWS
- La modalità automatica di EKS assegna automaticamente le autorizzazioni Kubernetes RBAC a questo ruolo utilizzando le voci di accesso EKS.
- AWS suggerisce di assegnare un nome a questo ruolo`AmazonEKSAutoNodeRole`.
- Con EKS Auto Mode, AWS suggerisce di creare un singolo ruolo IAM del nodo per AWS account.
- Questo ruolo dispone di autorizzazioni limitate. Le autorizzazioni chiave includono l'assunzione di un ruolo Pod Identity e l'estrazione di immagini da ECR.
- AWS suggerisce le seguenti politiche IAM AWS gestite:
  - `AmazonEKSWorkerNodeMinimalPolicy`
  - `AmazonEC2ContainerRegistryPullOnly`

Per ulteriori informazioni sul ruolo IAM del nodo e sulle politiche IAM AWS gestite, consulta:

- the section called " AWS politiche gestite"
- the section called "Ruolo IAM del nodo"

Per ulteriori informazioni sull'accesso a Kubernetes, consulta:

- the section called "Rivedere le policy di accesso"

### Service-linked ruolo

Amazon EKS utilizza un ruolo collegato ai servizi (SLR) per determinate operazioni. Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon EKS. Service-linked i ruoli sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

AWS crea e configura automaticamente la SLR. Puoi eliminare un SLR solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché non puoi rimuovere involontariamente delle autorizzazioni di accesso alle risorse.

La policy SLR concede ad Amazon EKS le autorizzazioni per osservare ed eliminare i componenti principali dell'infrastruttura: risorse EC2 (istanze, interfacce di rete, gruppi di sicurezza), risorse ELB (sistemi di bilanciamento del carico, gruppi target), CloudWatch funzionalità (registrazione e metriche) e ruoli IAM con prefisso «eks». Inoltre, consente il networking privato degli endpoint tramite l'associazione di zone e include le autorizzazioni per il monitoraggio e la pulizia delle risorse. VPC/ hosted EventBridge EKS-tagged

Per ulteriori informazioni, consulta:

- the section called "AWS policy gestita: AmazonEKSServiceRolePolicy"
- the section called "Service-linked autorizzazioni di ruolo per Amazon EKS"

### Riferimento alla policy di accesso

Per ulteriori informazioni sulle autorizzazioni di Kubernetes utilizzate dalla modalità automatica di EKS, consulta the section called "Rivedere le policy di accesso".

## Informazioni sulla rete VPC e sul bilanciamento del carico in modalità automatica di EKS

Questo argomento spiega come configurare le funzionalità di rete e bilanciamento del carico del cloud privato virtuale (VPC) in modalità automatica di EKS. Sebbene modalità automatica di EKS gestisca automaticamente la maggior parte dei componenti di rete, puoi comunque personalizzare

alcuni aspetti della configurazione di rete del cluster tramite risorse `NodeClass` e annotazioni del bilanciatore del carico.

Quando si utilizza EKS Auto Mode, AWS gestisce la configurazione VPC Container Network Interface (CNI) e il provisioning del load balancer per il cluster. Puoi influenzare i comportamenti di rete definendo oggetti `NodeClass` e applicando annotazioni specifiche alle risorse Service e Ingress, mantenendo al contempo il modello operativo automatizzato fornito dalla modalità automatica di EKS.

## Funzionalità di rete

modalità automatica di EKS ha una nuova funzionalità di rete che gestisce la rete di nodi e pod. Puoi configurarlo creando un oggetto `NodeClass` di Kubernetes.

Le opzioni di configurazione per il precedente AWS VPC CNI non si applicheranno alla modalità EKS Auto.

### Configura la rete con una **`NodeClass`**

La risorsa `NodeClass` in modalità automatica di EKS ti consente di personalizzare alcuni aspetti della funzionalità di rete. Tramite `NodeClass`, puoi specificare le selezioni dei gruppi di sicurezza, controllare il posizionamento dei nodi nelle sottoreti VPC, impostare le policy SNAT, configurare le policy di rete e abilitare la registrazione degli eventi di rete. Questo approccio mantiene il modello operativo automatizzato della modalità automatica di EKS fornendo al contempo flessibilità per la personalizzazione della rete.

Puoi usare una `NodeClass` per:

- Selezionare un gruppo di sicurezza per i nodi
- Controllare come vengono posizionati i nodi nelle sottoreti VPC
- Impostare la policy SNAT del nodo su `random` o `disabled`
- Abilitare le policy di rete Kubernetes, tra cui:
  - Impostare la policy di rete su Rifiuto per default o Autorizzazione per default
  - Abilitare la registrazione degli eventi di rete su un file.
- Isolare il traffico dei pod dal traffico del nodo collegando i pod a diverse sottoreti.

Scopri come creare un Amazon EKS. NodeClass

### Considerazioni

La modalità automatica di EKS supporta:

- Policy di rete EKS.
- Le opzioni `HostPort` e `HostNetwork` per i pod di Kubernetes.
- Nodi e pod in sottoreti pubbliche o private.
- Memorizzazione nella cache di query DNS sul nodo.
- Configurazione statica dell'interfaccia di rete per carichi di lavoro EFA. Per ulteriori informazioni, consulta the section called “Configurazione statica dell'interfaccia di rete”.

La modalità automatica di EKS non supporta:

- Gruppi di sicurezza per pod (SGPP). Per applicare gruppi di sicurezza separati al traffico Pod in modalità automatica, utilizza invece `podSecurityGroupSelectorTerms` in. `NodeClass` Per ulteriori informazioni, consulta the section called “Sottoreti e gruppi di sicurezza separati per i Pods”.
- Rete personalizzata nella `ENIConfig`. Puoi inserire i pod in più sottoreti o isolarli esclusivamente dal traffico dei nodi con the section called “Sottoreti e gruppi di sicurezza separati per i Pods”.
- Configurazioni warm IP, warm prefix e warm ENI.
- Configurazione minima delle destinazioni IP.
- Altre configurazioni supportate dal AWS VPC CNI open source.
- Configurazioni delle policy di rete come la personalizzazione del timer conntrack (l'impostazione predefinita è 300 s).
- Esportazione dei registri degli eventi di rete in. CloudWatch
- Plugin CNI alternativi (come Cilium o Calico) e plug-in alternativi per le politiche di rete. Amazon EKS Auto Mode utilizza la funzionalità di rete integrata descritta in precedenza. Per ulteriori informazioni, consulta Considerazioni sulla modalità Amazon EKS Auto.

### Gestione delle risorse di rete

EKS Auto Mode gestisce i prefissi, gli indirizzi IP e la gestione delle interfacce di rete monitorando NodeClass le risorse per le configurazioni di rete. Il servizio esegue automaticamente diverse operazioni sulle chiavi:

### Delega del prefisso

EKS Auto Mode utilizza per impostazione predefinita la delega dei prefissi (prefissi /28) per il networking dei pod e mantiene un warm pool predefinito di risorse IP scalabile in base al numero di pod pianificati. Quando viene rilevata la frammentazione della sottorete del pod, Auto Mode fornisce indirizzi IP secondari (/32). Grazie a questo algoritmo di rete pod predefinito, Auto Mode calcola il numero massimo di pod per nodo in base al numero di ENI e IP supportati per tipo di istanza (ipotizzando il caso peggiore di frammentazione). Per ulteriori informazioni sul numero massimo di ENI e IP per tipo di istanza, consulta Numero massimo di indirizzi IP per interfaccia di rete nella Guida per l'utente EC2. Le famiglie di istanze di nuova generazione (Nitro v6 e versioni successive) hanno generalmente un numero maggiore di ENI e IP per tipo di istanza e la modalità automatica regola di conseguenza il calcolo del numero massimo di pod.

Per i cluster IPv6, viene utilizzata solo la delega dei prefissi. La modalità automatica applica lo stesso IPv4-based calcolo e limita il numero massimo di pod a 110 per nodo, utilizzando il valore calcolato se è inferiore a 110.

### Gestione del tempo di raffreddamento

Il servizio implementa un pool di raffreddamento per prefissi o indirizzi IPv4 secondari che non sono più in uso. Dopo la scadenza del tempo di raffreddamento, queste risorse vengono rilasciate nuovamente al VPC. Tuttavia, se i pod riutilizzano queste risorse durante il tempo di raffreddamento, vengono ripristinate dal pool di raffreddamento.

### Supporto IPv6

Per i cluster IPv6, modalità automatica di EKS fornisce un prefisso `/80` IPv6 per nodo sull'interfaccia di rete principale. Quando si utilizza`podSubnetSelectorTerms`, il prefisso viene invece allocato su un'interfaccia di rete secondaria nella sottorete del pod.

Il servizio garantisce inoltre una corretta gestione e rimozione di oggetti inutili di tutte le interfacce di rete.

## Sistema di bilanciamento del carico

Si configurano gli AWS Elastic Load Balancer forniti da EKS Auto Mode utilizzando le annotazioni sulle risorse Service e Ingress.

Per ulteriori informazioni, consulta the section called “Creare una classe in ingresso” o the section called “Creare servizio”.

Considerazioni sul bilanciamento del carico con la modalità automatica di EKS

- La modalità di targeting predefinita è la modalità IP, non la modalità istanza.
- La modalità automatica di EKS supporta solo la modalità del gruppo di sicurezza per Network Load Balancer.
- AWS non supporta la migrazione dei load balancer dal controller di bilanciamento del carico autogestito alla gestione tramite EKS Auto AWS Mode.
- Il campo `networking.ingress.ipBlock` nella specifica `TargetGroupBinding` non è supportato.
- Se i nodi worker utilizzano gruppi di sicurezza personalizzati (non schemi di denominazione `eks-cluster-sg-` ), il ruolo del cluster richiede autorizzazioni IAM aggiuntive. La EKS-managed politica predefinita consente a EKS solo di modificare i gruppi di sicurezza denominati. `eks-cluster-sg-` Senza il permesso di modificare i gruppi di sicurezza personalizzati, EKS non può aggiungere le regole di ingresso richieste che consentono al ALB/NLB traffico di raggiungere i pod.

Considerazioni su CoreDNS

EKS Auto Mode non utilizza la tradizionale distribuzione di CoreDNS per fornire la risoluzione DNS all'interno del cluster. Invece, i nodi Auto Mode utilizzano CoreDNS in esecuzione come servizio di sistema direttamente su ciascun nodo. Se si esegue la transizione di un cluster tradizionale alla modalità automatica, è possibile rimuovere la distribuzione CoreDNS dal cluster una volta spostati i carichi di lavoro nei nodi Auto Mode.

> **Important**
>
> Se prevedi di mantenere un cluster con nodi Auto Mode e non Auto Mode, devi mantenere la distribuzione CoreDNS. Non-Auto I nodi Mode si affidano ai tradizionali pod CoreDNS per la risoluzione DNS, in quanto non possono accedere al servizio DNS a livello di nodo fornito da Auto Mode.

# Osservabilità per la modalità automatica EKS

Usa questo capitolo per conoscere le opzioni di osservabilità per i cluster Amazon EKS Auto Mode.

Argomenti

- Accesso AWS-Log dei componenti gestiti per EKS Auto

# Accesso AWS-Log dei componenti gestiti per EKS Auto

È possibile accedere ai log dei componenti AWS gestiti da EKS Auto Mode per ottenere un'osservabilità più approfondita delle operazioni del cluster. EKS Auto Mode supporta i log per le seguenti fonti:

- Scalabilità automatica del calcolo - Karpenter
- Storage a blocchi - EBS CSI
- Bilanciamento del carico - Load Balancer Controller AWS
- Rete Pod - Gestione degli indirizzi IP CNI VPC

I log possono essere consegnati a una destinazione di consegna a tua scelta.

Quando si crea un cluster EKS Auto Mode, è possibile abilitare la registrazione del piano di controllo (server API, Audit, Authenticator, Controller manager, Scheduler). I log dei componenti gestiti da EKS Auto (come Compute, Block storage, Load Balancing e IPAM) richiedono una configurazione separata tramite la consegna dei log.

## Impostazione della consegna dei log

Per configurare la distribuzione dei log dei componenti AWS gestiti per il tuo cluster EKS Auto Mode, utilizza l'API Amazon Logs. CloudWatch Per istruzioni dettagliate sulla configurazione, consulta Enabling logging from AWS services nella Amazon CloudWatch Logs User Guide. Ogni funzionalità della modalità automatica può essere configurata come una singola fonte di distribuzione di CloudWatch Vended Logs, consentendoti di selezionare i log a cui desideri avere accesso.

EKS Auto Mode supporta i seguenti tipi di log:

- AUTO_MODE_COMPUTE_LOGS
- AUTO_MODE_BLOCK_STORAGE_LOGS
- AUTO_MODE_LOAD_BALANCING_LOGS
- AUTO_MODE_IPAM_LOGS

Utilizzo delle API Amazon CloudWatch

La configurazione della registrazione richiede tre passaggi:

1. Crea una fonte di distribuzione per la funzionalità utilizzando l'API CloudWatch PutDeliverySource

2. Crea una destinazione di consegna utilizzando PutDeliveryDestination
3. Crea una consegna per collegare l'origine e la destinazione utilizzando CreateDelivery

È possibile configurare i dettagli della destinazione per i log di Auto Mode utilizzando l'DestinationConfiguration oggetto di consegna nell' CloudWatch PutDeliveryDestination API. Richiede l'ARN di un gruppo di CloudWatch log, di un bucket S3 o di un flusso di distribuzione di Amazon Data Firehose.

Puoi configurare una singola funzionalità Auto Mode (origine di consegna) per inviare i log a più destinazioni creando più consegne. Puoi anche creare più consegne per configurare più fonti di consegna per inviare i log alla stessa destinazione di consegna.

autorizzazioni IAM

A seconda della destinazione selezionata, potrebbe essere necessario configurare le policy o i ruoli IAM per il gruppo di CloudWatch log, il bucket S3 e Amazon Data Firehose per garantire la corretta consegna dei log. Inoltre, se stai inviando log tra AWS account, dovrai utilizzare l' PutDeliveryDestinationPolicy API per configurare una policy IAM che consenta la consegna a destinazione. Per ulteriori informazioni, consulta la documentazione sulle autorizzazioni di CloudWatch Vended Logs.

## Visualizzazione dei log

Una volta configurata la consegna dei log, i log verranno recapitati alla destinazione specificata. Il metodo per accedere ai log dipende dal tipo di destinazione scelto:

- CloudWatch Registri: visualizza i CloudWatch log nella console Logs, utilizza i comandi AWS CLI o esegui una query con CloudWatch Logs Insights
- Amazon S3: accedi ai log come oggetti nel bucket S3 tramite la console S3, la AWS CLI o strumenti di analisi come Amazon Athena
- Amazon Data Firehose: i log vengono trasmessi al target Firehose configurato (ad esempio S3, Service, Redshift, ecc.) OpenSearch

## Prezzi

CloudWatch Le tariffe di Vended Logs si applicano per la consegna e l'archiviazione dei log in base alla destinazione di consegna scelta. CloudWatch Vended Logs consente una consegna affidabile

e sicura dei log con AWS autenticazione e autorizzazione integrate a un prezzo ridotto rispetto ai log standard CloudWatch . Per maggiori dettagli, consulta la sezione Vended Logs della pagina CloudWatch dei prezzi.

Risorse correlate

- Registrazione del piano di controllo Amazon EKS
- PutDeliverySource API nel CloudWatch Logs API Reference
- PutDeliveryDestination API nel CloudWatch Logs API Reference
- CreateDelivery API nel CloudWatch Logs API Reference

# Risoluzione dei problemi della modalità automatica di EKS

Con la modalità automatica EKS, AWS si assume una maggiore responsabilità per le istanze EC2 nel tuo account. AWS EKS si assume la responsabilità del runtime dei container sui nodi, del sistema operativo sui nodi e di determinati controller. Ciò include un controller per l'archiviazione a blocchi, uno per il bilanciamento del carico e uno di elaborazione.

È necessario utilizzare le API AWS Kubernetes per risolvere i problemi dei nodi. Puoi:

- Utilizzare una risorsa `NodeDiagnostic` di Kubernetes per recuperare i log dei nodi utilizzando the section called "Agente di monitoraggio del nodo". Per le altri fasi, consulta the section called "Ottenere i log dei nodi".
- Usa una risorsa Kubernetes per acquisire il traffico di rete su `NodeDiagnostic` un nodo. Per le altri fasi, consulta the section called "Cattura il traffico di rete".
- Usa il comando CLI AWS EC2 `get-console-output` per recuperare l'output della console dai nodi. Per le altri fasi, consulta the section called "Ottieni l'output della console da un'istanza gestita da EC2 utilizzando AWS CLI EC2".
- Usa i container di debug di Kubernetes per recuperare i log dei nodi. Per le altri fasi, consulta the section called "Ottieni i log dei nodi usando contenitori di debug e la CLI di `kubectl`".

> **Note**
>
> modalità automatica di EKS utilizza istanze gestite EC2. Non puoi accedere direttamente alle istanze gestite EC2, anche tramite SSH.

Potresti riscontrare i seguenti problemi che hanno soluzioni specifiche per i componenti della modalità automatica di EKS:

- Pod bloccati nello stato `Pending`, che non vengono pianificati sui nodi Auto Mode. Per le soluzioni, consulta the section called "Risolvere i problemi relativi alla mancata pianificazione dei Pod sul nodo Auto Mode".
- Istanze gestite da EC2 che non entrano a far parte del cluster come nodi di Kubernetes. Per le soluzioni, consulta the section called "Risolvere i problemi del nodo che non si unisce al cluster".
- Errori e problemi con i `NodePools`, `PersistentVolumes` e `Services` che utilizzano i controller inclusi in modalità automatica di EKS. Per le soluzioni, consulta the section called "Risolvere i problemi relativi ai controller inclusi in Auto Mode".
- La sicurezza avanzata dei Pod impedisce la condivisione di volumi tra Pod. Per le soluzioni, consulta the section called "Condivisione di volumi tra pod".

Puoi utilizzare i seguenti metodi per risolvere i problemi relativi ai componenti della modalità automatica di EKS:

- the section called "Ottieni l'output della console da un'istanza gestita da EC2 utilizzando AWS CLI EC2"
- the section called "Ottieni i log dei nodi usando contenitori di debug e la CLI di `kubectl`"
- the section called "Visualizza le risorse associate a EKS Auto Mode nel AWS Console"
- the section called "Visualizza gli errori IAM nel tuo AWS account"
- the section called "Rileva i problemi di connettività dei nodi con `VPC Reachability Analyzer`"

## Agente di monitoraggio del nodo

modalità automatica di EKS include l'agente di monitoraggio dei nodi di Amazon EKS. Puoi utilizzare questo agente per visualizzare le informazioni di risoluzione dei problemi e di debug relative ai nodi. L'agente di monitoraggio dei nodi pubblica gli `events` di Kubernetes e le `conditions` del nodo. Per ulteriori informazioni, consulta the section called "Salute e riparazione dei nodi".

## Ottieni l'output della console da un'istanza gestita da EC2 utilizzando AWS CLI EC2

Questa procedura aiuta a risolvere problemi in fase di avvio o a livello di kernel.

Innanzitutto, devi determinare l'ID dell'istanza EC2 dell'istanza associata al carico di lavoro. In secondo luogo, usa la AWS CLI per recuperare l'output della console.

1. Conferma di avere `kubectl` installato e connesso al cluster
2. (Facoltativo) Utilizza il nome di un'implementazione di Kubernetes per elencare i pod associati.

```
kubectl get pods -l app=<deployment-name>
```

3. Utilizza il nome del pod di Kubernetes per determinare l'ID dell'istanza EC2 del nodo associato.

```
kubectl get pod <pod-name> -o wide
```

4. Utilizza l'ID dell'istanza EC2 per recuperare l'output della console.

```
aws ec2 get-console-output --instance-id <instance id> --latest --output text
```

## Ottieni i log dei nodi usando contenitori di debug e la CLI di **`kubectl`**

Il metodo consigliato per recuperare i log da un nodo EKS Auto Mode consiste nell'utilizzare una risorsa. `NodeDiagnostic` Per la procedura, consulta the section called "Ottenere i log dei nodi".

Tuttavia, puoi eseguire lo streaming dei log in tempo reale da un'istanza utilizzando il comando `kubectl debug node`. Questo comando avvia un nuovo Pod sul nodo di cui desideri eseguire il debug, che puoi quindi utilizzare in modo interattivo.

1. Avvia un container di debug. Il comando seguente utilizza `i-0123456789012345 6` per l'ID di istanza del nodo, `-it` alloca a `tty` e allega `stdin` per l'utilizzo interattivo, e utilizza il profilo `sysadmin` del file kubeconfig.

```
kubectl debug node/i-0123456789012345 6 -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023
```

   Di seguito viene riportato un output di esempio.

```
Creating debugging pod node-debugger-i-0123456789012345 6-nxb9c with container
 debugger on node i-0123456789012345 6.
If you don't see a command prompt, try pressing enter.
bash-5.2#
```

2. Dalla shell, puoi ora installare `util-linux-core` che fornisce il comando `nsenter`. Utilizza `nsenter` per inserire il namespace di montaggio di PID 1 (`init`) sull'host ed eseguire il comando `journalctl` per eseguire lo streaming dei log da `kubelet`:

```
yum install -y util-linux-core
nsenter -t 1 -m journalctl -f -u kubelet
```

Per motivi di sicurezza, l'immagine del container di Amazon Linux non installa molti file binari per impostazione predefinita. Puoi utilizzare il comando `yum whatprovides` per identificare il pacchetto da installare al fine di fornire un determinato file binario.

```
yum whatprovides ps
```

```
Last metadata expiration check: 0:03:36 ago on Thu Jan 16 14:49:17 2025.
procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : @System
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps

procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : amazonlinux
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps
```

## Visualizza le risorse associate a EKS Auto Mode nel AWS Console

È possibile utilizzare la AWS console per visualizzare lo stato delle risorse associate al cluster EKS Auto Mode.

- Volumi EBS
  - Visualizza i volumi della modalità automatica di EKS cercando la chiave del tag `eks:eks-cluster-name`
- Sistemi di bilanciamento del carico
  - Visualizza i bilanciatori del carico della modalità automatica di EKS cercando la chiave del tag `eks:eks-cluster-name`
- Istanze EC2

Le istanze EKS Auto Mode sono istanze gestite da EC2. A partire dal 22 aprile 2026, le nuove istanze gestite e le risorse associate sono nascoste per impostazione predefinita dalle visualizzazioni della console EC2 e dalle operazioni degli elenchi API (ad esempio). `DescribeInstances` Se non vedi le istanze Auto Mode nella console EC2, controlla le impostazioni di visibilità delle risorse gestite:

1. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel pannello di navigazione seleziona Pannello di controllo.
3. Nella scheda Attributi dell'account, in Impostazioni, scegli Visibilità gestita delle risorse.
4. Scegli Gestisci, quindi seleziona l'interruttore di visibilità per le istanze gestite.
5. Scegli Save changes (Salva modifiche).

Dopo aver abilitato la visibilità, puoi visualizzare le istanze EKS Auto Mode cercando la chiave del tag. `eks:eks-cluster-name`

In alternativa, è possibile visualizzare i nodi Auto Mode tramite:

- La console Amazon EKS, nella scheda Compute del cluster.
- `kubectl get nodes`da un client Kubernetes configurato.
- Interrogazioni dirette all'API EC2 tramite ID di istanza o con il parametro. `include-managed-resources`

Per ulteriori informazioni, consulta le impostazioni di visibilità delle risorse gestite nella Amazon EC2 User Guide.

## Visualizza gli errori IAM nel tuo AWS account

1. Accedi alla CloudTrail console.
2. Seleziona Cronologia eventi dal riquadro di navigazione a sinistra.
3. Scegli l'icona delle impostazioni (ingranaggio) in alto a destra nella tabella Cronologia eventi e abilita la colonna Codice di errore.
4. Controlla la colonna Codice di errore per individuare eventuali errori relativi alle autorizzazioni, ad esempio`AccessDenied`, `UnauthorizedOperation` e. `InvalidClientTokenId` Per restringere l'elenco, filtra in base all'origine dell'evento (ad esempio, `ec2.amazonaws.com` o`eks.amazonaws.com`) e all'intervallo di tempo.

Per un filtraggio più preciso in base al codice di errore in una finestra temporale più ampia, interroga un data store di eventi di CloudTrail Lake o crea una tabella Athena (le opzioni della tabella Query in Lake e Create Athena nella parte superiore della pagina Cronologia eventi).

Cerca gli errori relativi al cluster EKS. Utilizza i messaggi di errore per aggiornare le voci di accesso EKS, il ruolo IAM del cluster o il ruolo IAM del nodo. Potresti dover allegare una nuova policy a questi ruoli con autorizzazioni per la modalità automatica di EKS.

## Risolvere i problemi relativi alla mancata pianificazione dei Pod sul nodo Auto Mode

Se i pod rimangono nello `Pending` stato e non vengono pianificati su un nodo in modalità automatica, verifica se il pod o il manifesto di distribuzione ha un. `nodeSelector` Se è presente un `nodeSelector`, assicurati che utilizzi `eks.amazonaws.com/compute-type: auto` per essere pianificato sui nodi creati dalla modalità automatica di EKS. Per ulteriori informazioni sulle etichette dei nodi utilizzate dalla modalità automatica di EKS, consulta the section called "Controllare l'implementazione".

## Risolvere i problemi del nodo che non si unisce al cluster

La modalità automatica di EKS configura automaticamente le nuove istanze EC2 con le informazioni corrette per unirsi al cluster, inclusi l'endpoint del cluster e l'autorità di certificazione (CA) del cluster. Tuttavia, queste istanze possono ancora non riuscire a unirsi al cluster EKS sotto forma di nodo. Esegui i seguenti comandi per identificare le istanze che non si sono unite al cluster:

1. Esegui `kubectl get nodeclaim` per verificare le `NodeClaims` che sono `Ready = False`.

```
kubectl get nodeclaim
```

2. Esegui `kubectl describe nodeclaim <node_claim>` e cerca su Stato per trovare eventuali problemi che impediscono al nodo di unirsi al cluster.

```
kubectl describe nodeclaim <node_claim>
```

Messaggi di errore comuni:

`Error getting launch template configs`

Potresti ricevere questo errore se stai impostando tag personalizzati nella `NodeClass` con le autorizzazioni del ruolo IAM del cluster predefinite. Consulta the section called "Identità e accesso".

`Error creating fleet`

Potrebbe esserci qualche problema di autorizzazione con la chiamata `RunInstances` dall'API EC2. Verifica AWS CloudTrail la presenza di errori e verifica the section called "Ruolo IAM del cluster Auto Mode" le autorizzazioni IAM richieste.

## Rileva i problemi di connettività dei nodi con **`VPC Reachability Analyzer`**

> **Note**
>
> Ti viene addebitato un costo per ogni analisi eseguita da VPC Reachability Analyzer. Per i dettagli sui prezzi, consulta Prezzi di Amazon VPC.

Un motivo per cui un'istanza non si è unita al cluster è un problema di connettività di rete che impedisce di raggiungere il server API. Per diagnosticare questo problema, puoi utilizzare VPC Reachability Analyzer per eseguire un'analisi della connettività tra un nodo che non riesce a unirsi al cluster e al server API. Avrai bisogno di due informazioni:

- ID di istanza di un nodo che non è in grado di unirsi al cluster
- Indirizzo IP dell'endpoint del server dell'API Kubernetes

Per ottenere l'ID dell'istanza, dovrai creare un carico di lavoro sul cluster per far sì che modalità automatica di EKS avvii un'istanza EC2. Questo crea anche un oggetto `NodeClaim` nel cluster che avrà l'ID dell'istanza. Esegui `kubectl get nodeclaim -o yaml` per stampare tutte le `NodeClaims` del cluster. Ciascuna `NodeClaim` contiene l'ID dell'istanza come campo e di nuovo nel providerID:

```
kubectl get nodeclaim -o yaml
```

Di seguito viene riportato un output di esempio.

```
    nodeName: i-01234567890123456
    providerID: aws:///us-west-2a/i-01234567890123456
```

Puoi determinare l'endpoint del server dell'API Kubernetes eseguendo `kubectl get endpoint kubernetes -o yaml`. Gli indirizzi si trovano nel campo degli indirizzi:

```
kubectl get endpoints kubernetes -o yaml
```

Di seguito viene riportato un output di esempio.

```
apiVersion: v1
kind: Endpoints
metadata:
  name: kubernetes
  namespace: default
subsets:
- addresses:
  - ip: 10.0.143.233
  - ip: 10.0.152.17
  ports:
  - name: https
    port: 443
    protocol: TCP
```

Con queste due informazioni, è possibile eseguire l'analisi. Per prima cosa, accedi a VPC Reachability Analyzer in Console di gestione AWS.

1. Scegli «Crea e analizza percorso»
2. Fornisci un nome per l'analisi (ad esempio "Errore nell'unione del nodo")
3. Per "Tipo di origine", seleziona "Istanze"
4. Inserisci l'ID dell'istanza del nodo soggetto a errore come "Origine"
5. Per la "Destinazione del percorso", seleziona "Indirizzo IP"
6. Inserisci uno degli indirizzi IP per il server API come "Indirizzo di destinazione"
7. Espandi la "Sezione di configurazione aggiuntiva dell'intestazione dei pacchetti"
8. Inserisci una "Porta di destinazione" di 443
9. Seleziona "Protocollo" come TCP se non è già selezionato

10. Scegli «Crea e analizza percorso»
11. L'esecuzione dell'analisi potrebbe richiedere alcuni minuti. Se i risultati dell'analisi indicano una raggiungibilità non riuscita, indicheranno dove si trovava l'errore nel percorso di rete in modo da poter risolvere il problema.

## Condivisione di volumi tra pod

I nodi della modalità automatica di EKS sono configurati con SELinux in modalità di applicazione che fornisce un maggiore isolamento tra i Pod in esecuzione sullo stesso nodo. Quando SELinux è abilitato, alla maggior parte dei pod senza privilegi verrà applicata automaticamente la propria etichetta di sicurezza multicategoria (MCS). Questa etichetta MCS è unica per ogni Pod ed è progettata per garantire che un processo in un Pod non possa manipolare uno in un altro Pod o sull'host. Anche se un Pod etichettato viene eseguito come root e ha accesso al filesystem host, non sarà in grado di manipolare i file, effettuare chiamate di sistema sensibili sull'host, accedere al runtime del container oppure ottenere il materiale delle chiavi segrete di kubelet.

Per questo motivo, potresti riscontrare problemi durante il tentativo di condividere dati tra Pod. Ad esempio, una `PersistentVolumeClaim` con una modalità di accesso di `ReadWriteOnce` non consentirà comunque a più Pod di accedere contemporaneamente al volume.

Per abilitare questa condivisione tra i Pod, puoi usare i Pod `seLinuxOptions` per configurare la stessa etichetta MCS su quei Pod. In questo esempio, assegniamo le tre categorie `c123,c456,c789` al Pod. Ciò non entrerà in conflitto con le categorie assegnate automaticamente ai Pod sul nodo, poiché a esse verranno assegnate solo due categorie.

```
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456,c789"
```

## Visualizzare gli eventi di Karpenter nei log del piano di controllo

Per i cluster EKS con i log del piano di controllo abilitati, puoi ottenere informazioni sulle azioni e sul processo decisionale di Karpenter interrogando i log. Ciò può essere particolarmente utile per risolvere i problemi della modalità automatica di EKS relativi al provisioning, al dimensionamento e alla terminazione dei nodi. Per visualizzare Karpenter-related gli eventi, utilizza la seguente query CloudWatch Logs Insights:

```
fields @timestamp, @message
```

```
| filter @logStream like /kube-apiserver-audit/
| filter @message like 'DisruptionBlocked'
or @message like 'DisruptionLaunching'
or @message like 'DisruptionTerminating'
or @message like 'DisruptionWaitingReadiness'
or @message like 'Unconsolidatable'
or @message like 'FailedScheduling'
or @message like 'NoCompatibleInstanceTypes'
or @message like 'NodeRepairBlocked'
or @message like 'Disrupted'
or @message like 'Evicted'
or @message like 'FailedDraining'
or @message like 'TerminationGracePeriodExpiring'
or @message like 'TerminationFailed'
or @message like 'FailedConsistencyCheck'
or @message like 'InsufficientCapacityError'
or @message like 'UnregisteredTaintMissing'
or @message like 'NodeClassNotReady'
| sort @timestamp desc
```

Questa query filtra Karpenter-related eventi specifici nei log di controllo di kube-apiserver. Gli eventi includono vari stati di interruzione, errori di pianificazione, problemi di capacità e problemi relativi ai nodi. Analizzando questi log, puoi comprendere meglio:

- Perché Karpenter intraprende determinate azioni.
- Qualsiasi problema che impedisca il corretto provisioning, il dimensionamento o la terminazione dei nodi.
- Potenziali problemi di capacità o compatibilità con i tipi di istanze.
- Eventi del ciclo di vita dei nodi come interruzioni, espulsioni o terminazioni.

Per utilizzare questa query:

1. Passa alla console CloudWatch
2. Seleziona "Logs Insights" dal riquadro di navigazione a sinistra
3. Scegli il gruppo di log per i log del piano di controllo del cluster EKS
4. Incolla la query nell'editor di query
5. Regola l'intervallo di tempo in base alle esigenze
6. Eseguire la query

I risultati mostrano una cronologia degli Karpenter-related eventi, che ti aiuta a risolvere i problemi e a comprendere il comportamento di EKS Auto Mode nel tuo cluster. Per esaminare le azioni di Karpenter su un nodo specifico, puoi aggiungere il seguente filtro di riga che specifica l'ID dell'istanza alla query:

```
|filter @message like /i-12345678910123456/
```

> **Note**
>
> Per utilizzare questa query, è necessario abilitare la registrazione del piano di controllo sul cluster EKS. Se non hai ancora eseguito questa operazione, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

## Risolvere i problemi relativi ai controller inclusi in Auto Mode

Se hai un problema con un controller, dovresti cercare:

- Se le risorse associate a quel controller sono valide e formattate correttamente.
- Se le risorse AWS IAM e Kubernetes RBAC sono configurate correttamente per il tuo cluster. Per ulteriori informazioni, consulta the section called "Identità e accesso".

## Risorse correlate

Usa questi articoli di re:POST per le procedure avanzate di AWS risoluzione dei problemi:

- Come risolvere i problemi di scalabilità più comuni in EKS? Auto-Mode
- Come posso risolvere i problemi di provisioning di nodepool e nodeclass personalizzati nella modalità automatica di Amazon EKS?
- Come posso risolvere i pool di nodi integrati in modalità automatica di EKS con stato sconosciuto?

# Rivedere le note di rilascio della modalità automatica di EKS

Questa pagina documenta gli aggiornamenti alla modalità automatica di Amazon EKS. Puoi controllare periodicamente questa pagina per annunci su funzionalità, correzioni di bug, problemi noti e funzionalità obsolete.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/automode/auto-change.adoc.atom
```

## 17 agosto 2026

Documentazione: sono state aggiunte istruzioni per concedere ai controller gestiti EKS Auto Mode un RBAC Kubernetes aggiuntivo tramite il gruppo sulla voce di accesso creata automaticamente. `eks:managed` `AWSServiceRoleForAmazonEKS` Ciò sblocca casi d'uso come la concessione al controller di bilanciamento del carico dell'accesso a uno specifico in modo che possa risolvere la configurazione OIDC su un ALB. `Secret` `Ingress` Per ulteriori informazioni, consulta la procedura dettagliata all'indirizzo. the section called "Concedi RBAC ai controller gestiti" the section called "Concedi l'accesso segreto a LBC"

## 12 agosto 2026

Funzionalità: il controller di bilanciamento del carico Amazon EKS Auto Mode ora supporta le funzionalità di Load AWS Balancer Controller fino alla versione 3.4. Tieni presente che l'API Gateway non è ancora supportata.

- Convalida JWT per Ingress: convalida i JSON Web Tokens (JWT) a livello di regola del listener Application Load Balancer (ALB) prima che le richieste raggiungano il backend, tramite una nuova azione. Ingress-level Per ulteriori informazioni, vedi Application Load Balancer ora supporta il flusso delle credenziali del cliente con la verifica JWT.
- Gruppi target ponderati per Network Load Balancer (NLB): distribuisci il traffico tra più gruppi target in base al peso su un servizio NLB. Questo supporta modelli di distribuzione blu-verdi e canarini, incluso un peso pari a 0 quando almeno un altro gruppo target ha un peso diverso da zero. Per ulteriori informazioni, consulta Network Load Balancer now support Weighted Target Groups.
- TargetGroupBinding eventi di riconciliazione: Amazon EKS ora emette eventi Kubernetes in caso di errori di TargetGroupBinding riconciliazione, rendendo visibili i problemi di registrazione degli obiettivi anziché solo attraverso i log dei controller. `kubectl describe`
- L'ordine delle sottoreti è preservato: l'`aws-load-balancer-subnets`annotazione ora rispetta l'ordine specificato, anziché riordinare le sottoreti internamente.
- Cross-zone bilanciamento del carico per ALB: ora puoi disabilitare esplicitamente il bilanciamento del carico tra zone su Application Load Balancer.

## 5 agosto 2026

Funzionalità: il controller di bilanciamento del carico Amazon EKS Auto Mode ora supporta gruppi target multi-cluster, corrispondenti al comportamento del AWS Load Balancer Controller. Con questa funzionalità, puoi condividere lo stesso ARN del gruppo target su più `TargetGroupBinding` risorse, in modo che un singolo gruppo target possa servire più cluster Kubernetes (nello stesso VPC) o accettare target da altre fonti. Per ulteriori informazioni, consulta the section called "Multi-cluster gruppi target".

## 27 luglio 2026

Funzionalità: il controller di bilanciamento del carico in modalità automatica di Amazon Elastic Kubernetes Service (Amazon EKS) ora supporta le funzionalità di Load Balancer Controller v2.13 e AWS v2.14.

- Riscrittura degli URL dell'Application Load Balancer (ALB): ora puoi trasformare gli URL delle richieste e gli header degli host prima che le richieste raggiungano i servizi di backend, senza modificare l'applicazione. Per ulteriori informazioni, consulta Introducing URL and host header rewrite with Application Load Balancers. AWS
- PrefixListsIDs e LoadBalancerName in IngressClassParams : ora puoi impostare gli elenchi di prefissi dei gruppi di sicurezza e un nome di bilanciamento del carico personalizzato per il tuo Application Load Balancer. Entrambi sono supportati come annotazioni Ingress e come campi in IngressClassParams (PrefixListSids e load). BalancerName Se impostata IngressClassParams, la configurazione si applica a tutti gli Ingress in. IngressClass Non è più necessario annotare ogni risorsa Ingress singolarmente.
- Frontend NLB for Ingress: ora puoi posizionare un Network Load Balancer (NLB) davanti a un Application Load Balancer. Questo combina gli indirizzi IP statici NLB e le funzionalità di routing di livello 7 di ALB. AWS PrivateLink Abilita questa funzionalità con alb.ingress.kubernetes. io/ enableannotazione -frontend-nlb. Per ulteriori informazioni, vedere Application Load Target Group for Network Load Balancer Balancer-type .
- Supporto per listener TCP_UDP: i servizi NLB possono ora utilizzare i listener TCP_UDP, che consentono il traffico TCP e UDP sulla stessa porta. Abilita questa funzionalità con service.beta.kubernetes. io/awsannotazione -load-balancer-enable-tcp-udp-listener.
- Per-target-group protocollo proxy: ora puoi configurare gli header del Proxy Protocol v2 a livello di singolo gruppo di destinazione utilizzando service.beta.kubernetes. io/awsannotazione -load-balancer-proxy-protocol-per-target-group, anziché applicare la configurazione a tutti i gruppi target in modo uniforme.

- Campo targetType in: ora puoi impostare il tipo di destinazione predefinito (istanza o IP IngressClassParams ) direttamente in, eliminando la necessità di annotare ogni risorsa Ingress singolarmente. IngressClassParams
- Rilevamento delle sottoreti in base alla raggiungibilità: la selezione della sottorete non richiede più strettamente kubernetes. io/role tag. Il controller ora ricorre all'analisi della raggiungibilità basata sulla tabella di percorso quando i tag sono assenti. Il controller attualmente non supporta questo fallback per i load balancer con un tipo di indirizzo IP dualstack.
- Supporto IPv4 IP Address Manager (IPAM) per ALB: un ALB connesso a Internet può ora ricavare i propri indirizzi IPv4 pubblici da un pool IPAM di Amazon Virtual Private Cloud (Amazon VPC) anziché da intervalli di indirizzi gestiti. AWS Ciò fornisce blocchi di indirizzi IP prevedibili per gli elenchi di indirizzi consentiti. Specifica il pool con alb.ingress.kubernetes. io/ipamannotazione -ipv4-pool-id. Per ulteriori informazioni, consulta Semplificare l'assegnazione degli indirizzi IP pubblici di ALB con VPC IPAM.

Funzionalità: aggiunta la politica di `Balanced` consolidamento per EKS Auto Mode. NodePools L'impostazione assegna un `spec.disruption.consolidationPolicy: Balanced` punteggio a ciascuna azione di consolidamento soppesando i risparmi sui costi di elaborazione rispetto ai costi di interruzione. Salta le azioni in cui l'interruzione supera i risparmi. Se lo usi `WhenEmpty` oggi, puoi passare a questo per `Balanced` ottenere i risparmi sui costi del consolidamento. Se lo usi `WhenEmptyOrUnderutilized` oggi, puoi passare a questo metodo per eliminare le interruzioni dei pod e `Balanced` ottenere vantaggi marginali. `WhenEmptye` `WhenEmptyOrUnderutilized` sono immutati e quelli esistenti NodePools mantengono il loro comportamento attuale. Per ulteriori informazioni, consulta the section called “Creazione di pool di nodi” Disruption nella documentazione di Karpenter.

## 21 luglio 2026

Funzionalità: aggiunto il supporto per la configurazione statica dell'interfaccia di rete attiva NodeClass. È ora possibile configurare le interfacce di rete Elastic Fabric Adapter (EFA) utilizzando `advancedNetworking.networkInterfaces` il provisioning della capacità sia dinamico che statico, abilitando EFA-ready i nodi per carichi di lavoro di formazione e inferenza distribuiti. Per ulteriori informazioni, consulta the section called “Configurazione statica dell'interfaccia di rete”.

## 30 giugno 2026

Funzionalità: Il controller di bilanciamento del carico EKS Auto Mode ora supporta le funzionalità di AWS Load Balancer Controller v2.10, v2.11 e v2.12.

A partire dalla versione upstream 2.12.0:

- Gestione delle priorità delle regole del listener: il controller può ora impostare e riordinare in modo esplicito le priorità delle regole del listener, risolvendo i conflitti di ordinamento quando più regole Ingress hanno come target lo stesso listener

A partire dalla versione upstream 2.11.0:

- Prenotazione delle unità di carico LCU (Load Balancer Capacity Unit): ora è possibile riservare unità di capacità sia sugli Application Load Balancer (ALB) che sui Network Load Balancer (NLB), garantendo prestazioni prevedibili per carichi di lavoro con modelli di traffico noti

A partire dalla versione upstream 2.10.0:

- Protezione ALB Shield Advanced: le risorse ALB possono ora essere protette con AWS Shield Advanced tramite alb.ingress.kubernetes. io/shield-annotazione di protezione avanzata
- Personalizzane uno personalizzato TargetGroupBinding : ora puoi fare riferimento a gruppi target preesistenti non creati dal controller, consentendo l'integrazione con l'infrastruttura gestita esternamente
- Supporto UDP per NLB dual-stack su cluster IPv6: i servizi NLB su cluster IPv6 ora supportano i listener di protocollo UDP
- Attributi del listener ALB HTTP e HTTPS: controllo degli attributi a livello di listener (ad esempio, comportamento di routing, modifiche all'intestazione) tramite annotazioni Fine-grained

## Aggiornamento sulle politiche gestite

AWS ha aggiornato AmazonEKSServiceRolePolicy e AmazonEKSLoadBalancingPolicy supporta queste nuove funzionalità.

## Azione richiesta per i clienti che utilizzano politiche IAM personalizzate

Se stai fornendo una politica IAM personalizzata per il ruolo del cluster EKS Auto Mode invece di utilizzare AWS-managed AmazonEKSLoadBalancingPolicy, devi assicurarti che la tua policy includa le autorizzazioni elencate sopra. Il mancato aggiornamento della policy personalizzata comporterà errori di accesso negato durante l'utilizzo delle nuove funzionalità.

Per verificare la parità, confronta la tua politica personalizzata con la versione più recente di AmazonEKSLoadBalancingPolicy.

In particolare, assicurati che la tua polizza includa:

- elasticloadbalancing:ModifyCapacityReservation, elasticloadbalancing:, elasticloadbalancing: ed elasticloadbalancing: ModifyIpPools ModifyListenerAttributes SetRulePriorities
- ec2: DescribeIpamPools ed ec2: DescribeRouteTables
- scudo:CreateProtection, scudo: DeleteProtection e scudo: TagResource

## 9 giugno 2026

Funzionalità: aggiunto il monitoraggio del controllo dello stato dell'istanza alla modalità automatica EKS. Il controller di calcolo ora interroga EC2 `DescribeInstanceStatus` per rilevare gli eventi di manutenzione programmata e gli errori di controllo dello stato dell'istanza o del sistema, sostituendo automaticamente i nodi non integri.

## 4 giugno 2026

Documentazione: sono state aggiunte linee guida sul controllo dei costi di elaborazione in EKS Auto Mode, tra cui come funziona il consolidamento, cosa lo blocca e modelli consigliati per carichi di lavoro intensivi. Per ulteriori informazioni, vedere Ottimizzazione dei costi in modalità EKS Auto.

## 3 giugno 2026

Funzionalità: aggiunto il supporto per le prenotazioni di capacità interrompibile in modalità EKS Auto. Per ulteriori informazioni, consulta the section called "Controllare le prenotazioni della capacità".

## 5 maggio 2026

Funzionalità: aggiunto il supporto per i gruppi di collocamento EC2 in modalità EKS Auto. Per ulteriori informazioni, consulta the section called "Specifiche della classe di nodi".

## 10 aprile 2026

Nuovi tipi di istanza supportati: p6-b200, p6-b300, p5e, p5en, trn2, hpc8a, x8aedz, x8i. Per l'elenco completo delle istanze supportate, consulta. the section called "Istanze gestite"

## 2 aprile 2026

Compito: la validazione a NodeClass secco ora utilizzerà tipi di istanza selezionati dinamicamente in base a linked. NodePools

## 2 febbraio 2026

Funzionalità: aggiunto il supporto per disabilitare il traffico V4egress dai pod IPv6 nei cluster IPv6 EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Disabilita l'uscita IPv4 dai pod IPv6 nei cluster IPv6.".

## 19 dicembre 2025

Funzionalità: è stato aggiunto il supporto per la modalità IP secondaria che fornisce indirizzi IP secondari anziché prefissi ai nodi Auto. La modalità mantiene un IP secondario come minimalIPTarget e consente di risparmiare risorse IP per i clienti che non hanno bisogno di riscaldare altri IP o prefissi secondari. Per ulteriori informazioni, consulta the section called "Specifiche della classe di nodi" e the section called "Modalità IP secondaria per pod".

## 19 novembre 2025

Funzionalità: funzionalità di pull and unpack in parallelo Seekable OCI (SOCI) abilitate per le istanze delle famiglie G, P e Trn con storage NVMe locale. Il pull and unpack parallelo SOCI viene sempre utilizzato per queste famiglie di istanze con EKS Auto Mode e non sono necessarie modifiche alla configurazione per abilitarlo. Per ulteriori informazioni su SOCI, consulta il blog di lancio.

## 19 novembre 2025

Funzionalità: è stato aggiunto il supporto per i pool di nodi a capacità statica che mantengono un numero fisso di nodi. Per ulteriori informazioni, consulta the section called "Crea un pool di nodi statico".

## 23 ottobre 2025

Funzionalità: gli utenti con cluster nelle regioni degli Stati Uniti possono ora richiedere di utilizzare AMI compatibili con FIPS `spec.advancedSecurity.fips` specificandolo nella loro definizione. NodeClass

## 1° ottobre 2025

Funzionalità: EKS Auto Mode ora supporta la distribuzione di nodi nelle zone locali. AWS Per ulteriori informazioni, consulta the section called "Usa Local Zones".

## 30 settembre 2025

Funzionalità: è stato aggiunto il supporto per InstanceProfile NodeClass `spec.instanceProfile`, che si esclude a vicenda dal campo. `spec.role`

## 29 settembre 2025

DRA non è attualmente supportato da EKS Auto Mode.

## 10 settembre 2025

Compito: gli eventi generati dal controller Auto Mode Compute ora utilizzeranno il nome `eks-auto-mode/compute` anziché `karpenter`.

## 24 agosto 2025

Correzione di bug: i VPC che utilizzavano un set di opzioni DHCP con un nome di dominio personalizzato contenente lettere maiuscole impedivano ai nodi di unirsi al cluster a causa della generazione di un nome host non valido. Questo problema è stato risolto e i nomi di dominio con lettere maiuscole ora funzionano correttamente.

## 15 agosto 2025

Correzione di bug: il Pod Identity Agent ora ascolterà solo l'indirizzo locale di collegamento IPv4 in un cluster IPv4 EKS per evitare problemi in cui il pod non riesce a raggiungere l'indirizzo IPv6.

## 6 agosto 2025

Funzionalità: aggiunta una nuova configurazione NodeClass `spec.advancedNetworking.associatePublicIPAddress` che può essere utilizzata per impedire l'assegnazione di indirizzi IP pubblici ai nodi EKS Auto Mode

## 30 giugno 2025

Funzionalità: La modalità automatica NodeClass ora utilizza la chiave KMS personalizzata configurata per crittografare il volume principale di sola lettura dell'istanza, oltre al volume di dati.

read/write In precedenza, la chiave KMS personalizzata veniva utilizzata solo per crittografare il volume di dati.

## 20 giugno 2025

Funzionalità:  supporto per il controllo dell'implementazione dei carichi di lavoro negli On-Demand EC2 Capacity Reservations (ODCR). Ciò aggiunge la chiave opzionale `capacityReservationSelectorTerms` a NodeClass, che consente di controllare in modo esplicito quali ODCR utilizzano i carichi di lavoro. Per ulteriori informazioni, consulta the section called "Controllare le prenotazioni della capacità".

## 13 giugno 2025

Funzionalità: supporto per sottoreti dei pod separate nella `NodeClass`. Vengono aggiunte le chiavi facoltative `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` per impostare le sottoreti e i gruppi di sicurezza per i pod. Per ulteriori informazioni, consulta the section called "Sottoreti e gruppi di sicurezza separati per i Pods".

## 30 aprile 2025

Funzionalità: supporto per proxy di rete inoltrati nella `NodeClass`. Viene aggiunta la chiave facoltativa `advancedNetworking` per impostare il proxy HTTPS. Per ulteriori informazioni, consulta the section called "Specifiche della classe di nodi".

## 18 aprile 2025

Funzionalità: supporto per la risoluzione di domini .local (in genere riservati per Multicast DNS) tramite DNS unicast.

## 11 aprile 2025

Funzionalità: aggiunta di `certificateBundles` e `ephemeralStorage.kmsKeyID` a `NodeClass` Per ulteriori informazioni, consulta the section called "Specifiche della classe di nodi".

Funzionalità: velocità di estrazione delle immagini migliorata, in particolare per i tipi di istanze con archiviazione locale delle istanze che possono sfruttare la decompressione più rapida delle immagini.

Correzione bug: è  stata risolta una race condition che causava, a volte FailedCreatePodSandBox , un errore durante la composizione: dial tcp 127.0.0. 1:50051: connect: connection refused per i Pods che programmavano un nodo immediatamente all'avvio.

## 4 aprile 2025

Funzionalità: aumentare `registryPullQPS` da 5 a 25 e `registryBurst` da 10 a 50 per ridurre la limitazione dell'estrazione delle immagini applicata dal client (`Failed to pull image xyz: pull QPS exceeded`)

## 31 marzo 2025

Correzione di bug: risoluzione di un problema per cui se un Core DNS Pod è in esecuzione su un nodo in Auto Mode, le query DNS dai Pod sul nodo raggiungevano quel Core DNS Pod anziché il server DNS locale del nodo. Le query DNS dai pod su un nodo in Auto Mode andranno sempre al DNS locale del nodo.

## 21 marzo 2025

Correzione di bug: i nodi in Auto Mode ora risolvono `kube-dns.kube-system.svc.cluster.local` correttamente quando non vi è un servizio `kube-dns` installato nel cluster. Risolve il problema #2546. GitHub

## 14 marzo 2025

Funzionalità: uscita `IPv4` abilitata nei cluster `IPv6`. Il traffico `IPv4` in uscita dai cluster `IPv6` in Auto Mode verrà ora tradotto automaticamente all'indirizzo `v4` del nodo ENI primario.

# Ciclo di vita e configurazione del cluster di Amazon EKS

Questa sezione fornisce indicazioni approfondite sulla gestione dell'intero ciclo di vita dei cluster Kubernetes e sui diversi modi per configurarli. Tratta la creazione di cluster, il monitoraggio dell'integrità dei cluster, l'aggiornamento delle versioni di Kubernetes e l'eliminazione dei cluster. Include anche argomenti avanzati su come configurare l'accesso agli endpoint, il supporto enable/disable Windows, configurare cluster privati, implementare la scalabilità automatica e come migliorare la resilienza con lo spostamento zonale per il reindirizzamento del traffico nelle configurazioni Multi-AZ.

Argomenti


- Creare un cluster della modalità automatica di Amazon EKS
- Crea un cluster Amazon EKS.
- Piano di controllo predisposto di Amazon EKS
- Configurazione avanzata del piano di controllo Kubernetes
- Prepararsi agli aggiornamenti delle versioni di Kubernetes e risolvere i problemi di configurazione errata con gli approfondimenti sui cluster
- Aggiornamento del cluster esistente alla nuova versione di Kubernetes
- Ripristina un cluster a una versione precedente di Kubernetes
- Eliminazione di un cluster
- Endpoint del server API del cluster
- Implementazione dei nodi Windows su cluster EKS
- Disabilitare il supporto Windows
- Implementazione di cluster privati con accesso limitato a Internet
- Configurazione del routing in uscita dal piano di controllo
- Dimensionamento dell'elaborazione dei cluster con Karpenter e Cluster Autoscaler
- Ulteriori informazioni relative allo spostamento zonale in controller di ripristino delle applicazioni Amazon (ARC)
- Abilitazione dello spostamento zonale EKS per evitare zone di disponibilità compromesse

# Creare un cluster della modalità automatica di Amazon EKS

Questo argomento offre istruzioni dettagliate per creare un cluster della modalità automatica di Amazon EKS usando opzioni di configurazione avanzate. Vengono trattati i prerequisiti, le opzioni di rete e le configurazioni aggiuntive. Il processo comprende l'impostazione dei ruoli IAM, la configurazione delle impostazioni del cluster, la specificazione dei parametri di rete e la selezione dei componenti aggiuntivi. Gli utenti possono creare cluster utilizzando l'interfaccia a riga di comando Console di gestione AWS o la AWS CLI, con una guida dettagliata fornita per entrambi i metodi.

Gli utenti che desiderano una procedura di configurazione meno complessa possono fare riferimento alle seguenti istruzioni semplificate per la creazione di cluster:

- the section called "CLI di eksctl"
- the section called " AWS CLI"
- the section called "Console di gestione"

Questa guida alla configurazione avanzata è destinata agli utenti che necessitano di un controllo più granulare sulla configurazione del cluster della modalità automatica di EKS e che hanno familiarità con i concetti e i requisiti di Amazon EKS. Prima di iniziare la configurazione avanzata, assicurati di aver soddisfatto tutti i prerequisiti e di avere una conoscenza approfondita dei requisiti di rete e IAM per i cluster della modalità automatica di EKS.

La modalità automatica di EKS richiede autorizzazioni IAM aggiuntive. Per ulteriori informazioni, consulta:

- the section called "Ruoli IAM per cluster della modalità automatica di EKS"
- the section called "Identità e accesso"

> **Note**
>
> Se vuoi creare un cluster senza modalità automatica di EKS, consulta the section called "Creazione di un cluster ".
> In questo argomento viene trattata la configurazione avanzata. Se vuoi iniziare a usare modalità automatica di EKS, consulta the section called "Creazione di un cluster".

## Prerequisiti

- Un VPC esistente e sottoreti che soddisfano i requisiti di Amazon EKS. Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di approfondire le nozioni relative ai requisiti del VPC e delle sottoreti. Se non disponi di un VPC e di sottoreti, puoi crearli utilizzando un modello fornito da Amazon EKS. AWS CloudFormation
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version`. Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.
- Un IAM principale con le autorizzazioni per creare e modificare risorse EKS e IAM.

## Crea cluster - AWS console

1. Aprire la Console Amazon EKS.
2. Seleziona Aggiungi cluster e quindi Crea.
3. In Opzioni di configurazione, seleziona Configurazione personalizzata.
   - In questo argomento viene trattata la configurazione personalizzata. Per ulteriori informazioni sulla configurazione rapida, consulta the section called "Console di gestione".
4. Conferma che l'opzione Usa modalità automatica di EKS sia abilitata.
   - In questo argomento viene trattata la creazione di cluster con la modalità automatica di EKS. Per ulteriori informazioni sulla creazione di cluster senza modalità automatica di EKS, consulta the section called "Creazione di un cluster ".
5. Nella pagina Configura cluster compila i campi seguenti:
   - Nome: un nome per il cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.

- Ruolo IAM del cluster: scegli il ruolo IAM del cluster Amazon EKS che hai creato per consentire al piano di controllo Kubernetes di gestire AWS le risorse per tuo conto. Se non hai mai creato in precedenza un ruolo IAM del cluster per la modalità automatica di EKS, seleziona il pulsante Crea ruolo consigliato per creare il ruolo con le autorizzazioni richieste nella console IAM.
- Versione Kubernetes – la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra una versione precedente.
- Policy di aggiornamento: la policy della versione di Kubernetes che vuoi impostare per il tuo cluster. Se vuoi che il cluster venga eseguito solo su una versione di supporto standard, puoi scegliere Standard. Se vuoi che il tuo cluster acceda al supporto esteso alla fine del supporto standard per una versione, puoi scegliere Esteso. Se selezioni una versione di Kubernetes che attualmente è in supporto esteso, non puoi selezionare il supporto standard come opzione.

6. Nella sezione Auto Mode Compute della pagina di configurazione del cluster, compila i seguenti campi:
   - Pool di nodi: determina se desideri utilizzare i pool di nodi integrati. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".
   - Ruolo IAM del nodo: se abiliti un pool di nodi integrato, devi selezionare un ruolo IAM del nodo. La modalità automatica di EKS assegnerà tale ruolo ai nuovi nodi. Non puoi modificare questo valore dopo la creazione del cluster. Se non hai mai creato in precedenza un ruolo IAM del nodo per la modalità automatica di EKS, seleziona il pulsante Crea ruolo consigliato per creare il ruolo con le autorizzazioni richieste. Per ulteriori informazioni su questo ruolo, consulta the section called "Identità e accesso".
7. Nella sezione Accesso al cluster della pagina Configura cluster, inserisci i seguenti campi:
   - Accesso amministratore cluster Bootstrap: il creatore del cluster diventa automaticamente amministratore Kubernetes. Se vuoi disabilitare questa opzione, seleziona Impedisci l'accesso all'amministratore del cluster.
   - Modalità di autenticazione cluster: modalità automatica di EKS richiede voci di accesso EKS, la modalità di autenticazione API EKS. È possibile abilitare facoltativamente la modalità di `ConfigMap` autenticazione selezionando EKS API e ConfigMap.
8. Inserisci i campi rimanenti nella pagina di configurazione del cluster:
   - Crittografia dei segreti (Facoltativo): scegli di abilitare la crittografia dei segreti per i segreti Kubernetes tramite una chiave KMS. Puoi abilitare questa funzionalità anche dopo la creazione del cluster. Assicurati, tuttavia, di acquisire familiarità con le informazioni contenute nella sezione the section called "Abilitare la crittografia dei segreti".

- Spostamento zonale ARC: la modalità automatica EKS non supporta lo spostamento zonale ARC.
- Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Tagging delle risorse ".

  Quando hai finito con questa pagina, seleziona Avanti.

9. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:

- VPC: crea il cluster in un VPC esistente conforme ai requisiti su VPC di Amazon EKS. Prima di scegliere un VPC, ti consigliamo di acquisire familiarità con i requisiti e le considerazioni nella sezione the section called "Requisiti del VPC e delle sottoreti". Dopo la creazione del cluster, non potrai più modificare il VPC da utilizzare. Se l'elenco risulta vuoto, è necessario crearne uno prima. Per ulteriori informazioni, consulta the section called "Crea un VPC".
- Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. È necessario selezionarne almeno due.

  Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.

  Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS.

  Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.
- Scegli la famiglia di indirizzi IP del cluster: puoi scegliere IPv4 o IPv6.

  Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

- (Facoltativo) Scegli Configure Kubernetes Service IP address range (Configura l'intervallo di indirizzi IP del servizio Kubernetes) e specifica un intervallo **`IPv4`** del servizio.

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

  - Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
  - Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
  - Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

  Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`.

- Per Cluster endpoint access (Accesso all'endpoint del cluster), seleziona un'opzione. Puoi modificare questa opzione dopo aver creato il cluster. Prima di selezionare un'opzione non predefinita, assicurati di familiarizzare con le opzioni e le relative implicazioni. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

  Quando hai finito con questa pagina, seleziona Avanti.

10. (Facoltativo) Nella pagina Configura osservabilità, scegli le opzioni Parametri e Registrazione del piano di controllo da attivare. Per impostazione predefinita, i tipi di log sono disattivati.

    - Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Passaggio 1: attivazione delle metriche Prometheus".
    - Per ulteriori informazioni sulle opzioni Registrazione del piano di controllo, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".
    - Quando hai finito con questa pagina, seleziona Avanti.

11. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster. Puoi scegliere tutti i componenti aggiuntivi di Amazon EKS e AWS Marketplace di cui hai bisogno. Se i componenti aggiuntivi di AWS Marketplace che desideri installare non sono elencati, puoi scegliere la numerazione delle pagine per visualizzare ulteriori risultati di pagina o cercare i componenti aggiuntivi di AWS Marketplace disponibili inserendo del testo nella casella di ricerca. Puoi anche filtrare per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca. Quando crei un cluster, puoi

visualizzare, selezionare e installare tutti i componenti aggiuntivi che supportano EKS Pod Identity, come descritto in dettaglio in the section called "Pod Identity".

- La modalità automatica di EKS automatizza la funzionalità di alcuni componenti aggiuntivi. Se prevedi di distribuire EKS Managed Node Groups nel tuo cluster EKS Auto Mode, seleziona Amazon EKS  aggiuntivo Add-ons  ed esamina le opzioni. Potrebbe essere necessaria l'installazione di componenti aggiuntivi come CoredNS e kube-proxy. EKS installerà i componenti aggiuntivi presenti in questa sezione solo su nodi e gruppi di nodi autogestiti.
- Quando hai finito con questa pagina, seleziona Avanti.

12 Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona la versione da installare. Dopo la creazione del cluster, puoi eseguire in qualsiasi momento l'aggiornamento a una versione successiva.

Per i componenti aggiuntivi che supportano EKS Pod Identities, puoi utilizzare la console per generare automaticamente il ruolo con il nome, la policy AWS gestita e la politica di fiducia precompilati specificamente per il componente aggiuntivo. È possibile riutilizzare i ruoli esistenti o crearne di nuovi per i componenti aggiuntivi supportati Per i passaggi necessari per usare la console per creare ruoli per i componenti aggiuntivi che supportano EKS Pod Identity, consulta the section called "Crea componente aggiuntivo (AWS Console)". Se un componente aggiuntivo non supporta EKS Pod Identity, viene mostrato un messaggio con le istruzioni per usare la procedura guidata per creare i ruoli IAM per gli account di servizio (IRSA) dopo la creazione del cluster.

Puoi aggiornare la configurazione di ogni componente aggiuntivo dopo la creazione del cluster. Per ulteriori informazioni sulla configurazione dei componenti aggiuntivi, consulta the section called "Aggiornamento di un componente aggiuntivo". Quando hai finito con questa pagina, seleziona Avanti.

13 Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE.

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di

disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Il provisioning del cluster richiede diversi minuti.

## Crea cluster - AWS CLI

Le seguenti istruzioni sulle CLI riguardano la creazione di risorse IAM e la creazione del cluster.

### Creazione di un ruolo IAM per un cluster della modalità automatica di EKS

Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Passaggio 2: creazione del ruolo IAM

Usa la policy di attendibilità per creare il ruolo IAM del cluster:

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

Passaggio 3: annotazione del ruolo ARN

Recupera e salva l'ARN del nuovo ruolo per usarlo nelle fasi successive:

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti politiche AWS gestite al ruolo IAM del cluster per concedere le autorizzazioni necessarie:

AmazonEKSClusterPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicyV2:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicyV2
```

AmazonEKSLoadBalancingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

## Creazione di un ruolo IAM per un nodo modalità automatica di EKS

### Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `node-trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

### Passaggio 2: creare il ruolo IAM del nodo

Usa il file node-trust-policy.json del passaggio precedente per definire quali entità possono assumere il ruolo. Per creare il ruolo IAM del nodo, esegui il seguente comando:

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

### Passaggio 3: annotazione del ruolo ARN

Una volta creato il ruolo, recupera e salva l'ARN del ruolo IAM del nodo. Sarà necessario usare questo ARN nei passaggi successivi. Usa il seguente comando per ottenere l'ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

### Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti politiche AWS gestite al ruolo IAM del nodo per fornire le autorizzazioni necessarie:

AmazonEKSWorkerNodeMinimalPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

AmazonEC2ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

## Creazione di un cluster

1. Crea un cluster con il comando seguente. Prima di eseguire il comando, apporta le modifiche seguenti:
   - Sostituiscilo *region-code* con la AWS regione in cui desideri creare il cluster.
   - Sostituisci *my-cluster* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Sostituisci *1.30* con qualsiasi versione supportata da Amazon EKS.
   - Sostituiscilo *111122223333* con l'ID del tuo account
   - Se hai creato ruoli IAM con nomi diversi per i ruoli Cluster e Nodo, sostituisci gli ARN.
   - Sostituisci i valori di `subnetIds` con quelli in tuo possesso. Puoi inoltre aggiungere altri ID. Devi specificare almeno due ID sottorete,

     Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.
   - Se non vuoi specificare un ID del gruppo di sicurezza, rimuovi `,securityGroupIds=sg-<ExampleID1>` dal comando. Se vuoi specificare uno o più ID del gruppo di sicurezza, sostituisci i valori di `securityGroupIds` con quelli in tuo possesso. Puoi inoltre aggiungere altri ID.

     Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo

gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.

```
aws eks create-cluster \
  --region region-code \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --role-arn arn:aws: iam::111122223333:role/AmazonEKSAutoClusterRole \
  --resources-vpc-config '{"subnetIds": ["subnet-ExampleID1","subnet-ExampleID2"],
 "securityGroupIds": ["sg-ExampleID1"], "endpointPublicAccess": true,
 "endpointPrivateAccess": true}' \
  --compute-config '{"enabled": true, "nodeRoleArn": "arn:aws:
 iam::111122223333:role/AmazonEKSAutoNodeRole", "nodePools": ["general-purpose",
 "system"]}' \
  --kubernetes-network-config '{"elasticLoadBalancing": {"enabled": true}}' \
  --storage-config '{"blockStorage": {"enabled": true}}' \
  --access-config '{"authenticationMode": "API"}'
```

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Di seguito sono riportate le impostazioni facoltative da aggiungere al comando precedente, quando necessario. Puoi abilitare queste opzioni esclusivamente durante la creazione del cluster, non in seguito.

- Se desideri specificare da quale blocco `IPv4` Classless Inter-domain Routing (CIDR) Kubernetes assegna gli indirizzi IP del servizio, devi specificarlo aggiungendo il comando seguente. `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>`

L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

Il blocco CIDR deve soddisfare i seguenti requisiti:

- Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
- Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
- Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`.

- Se stai creando un cluster e vuoi che il cluster assegni gli indirizzi `IPv6` ai pod e ai servizi anziché a indirizzi `IPv4`, aggiungi `--kubernetes-network-config ipFamily=ipv6` al seguente comando.

Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

2. Il provisioning del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

## Fasi successive

- the section called "Accedere al cluster con kubectl"
- the section called "Voci di accesso"
- Abilitare la crittografia dei segreti per il cluster.

- Configurare la registrazione per il cluster.
- Aggiungi nodi al cluster.

# Crea un cluster Amazon EKS.

> **Note**
>
> Questo argomento riguarda la creazione di cluster Amazon EKS senza EKS Auto Mode. Per istruzioni dettagliate su come creare un cluster della modalità automatica di EKS, consulta the section called "Creare un cluster automatico".
> Per iniziare a usare modalità automatica di EKS, consulta the section called "Crea cluster (EKS Auto Mode)".

Questo argomento offre una panoramica delle opzioni disponibili e descrivere gli aspetti da considerare quando crei un cluster Amazon EKS. Se devi creare un cluster con la tua infrastruttura on-premises come elaborazione per i nodi, consulta the section called "Creazione di un cluster". Se è la prima volta che crei un cluster Amazon EKS, consigliamo di seguire le nostre guide in Nozioni di base. Le guide consentono di creare un cluster predefinito in modo semplice, senza approfondire tutte le opzioni disponibili.

## Prerequisiti

- Un VPC esistente e sottoreti che soddisfano i requisiti di Amazon EKS. Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di approfondire le nozioni relative ai requisiti del VPC e delle sottoreti. Se non disponi di un VPC e di sottoreti, puoi crearli utilizzando un modello fornito da Amazon EKS. AWS CloudFormation
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come`yum`,`apt-get`, o Homebrew per macOS sono spesso diverse versioni inferiori all'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e

configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata AWS CloudShell potrebbe inoltre essere inferiore a quella più recente. Per aggiornarla, vedi Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.

- Un principale IAM con autorizzazioni per `create` e `describe` un cluster Amazon EKS. Per ulteriori informazioni, consulta the section called “Creazione di un cluster Kubernetes locale su un Outpost” e the section called “Elencare o descrivere tutti i cluster”.

## Passaggio 1: crea un ruolo IAM del cluster

1. Se disponi già di un ruolo IAM del cluster o intendi creare il cluster con `eksctl`, ignora questo passaggio. Per impostazione predefinita, `eksctl` crea un ruolo per te.
2. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Crea il ruolo IAM del cluster Amazon EKS. Se necessario, inserisci il prefisso *`eks-cluster-role-trust-policy.json`* nel percorso del computer in cui hai eseguito la scrittura del file nella Passaggio precedente. Il comando associa il criterio di attendibilità creato nella Passaggio precedente al ruolo. Per creare un ruolo IAM, l'azione `iam:CreateRole` (autorizzazione) deve essere assegnata al principale IAM che sta creando il ruolo.

```
aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

4. Puoi assegnare la policy gestita da Amazon EKS o creare una policy personalizzata. Per conoscere le autorizzazioni minime che devi utilizzare nella policy personalizzata, consulta the section called “Ruolo IAM del cluster”.

Allega al ruolo la policy gestita da Amazon EKS, denominata AmazonEKSClusterPolicy. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
```

### Service-Linked Ruolo

Amazon EKS crea automaticamente un ruolo collegato al servizio chiamato. `AWSServiceRoleForAmazonEKS`

Ciò avviene in aggiunta al ruolo IAM del cluster. Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. Il ruolo consente ad Amazon EKS di gestire i cluster nell'account. Per ulteriori informazioni, consulta the section called "Ruolo del cluster".

Il principale IAM utilizzato per creare il cluster EKS deve disporre dell'autorizzazione per creare il ruolo collegato al servizio. Ciò include l'autorizzazione `iam:CreateServiceLinkedRole`.

Se il ruolo collegato al servizio non esiste già e il ruolo IAM corrente non dispone delle autorizzazioni sufficienti per crearlo, l'operazione di creazione del cluster avrà esito negativo.

## Fase 2: Creare un cluster

Puoi creare un cluster utilizzando:

- `eksctl`
- il Console di gestione AWS
- la AWS CLI

### Crea cluster - eksctl

1. È necessaria la versione `0.215.0` o successiva dello strumento a riga di `eksctl` comando installato sul dispositivo oppure AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

2. Crea un `IPv4` cluster Amazon EKS con la versione Kubernetes predefinita di Amazon EKS nella tua regione predefinita. AWS Prima di eseguire il comando, apporta le modifiche seguenti:
3. Sostituiscilo *`region-code`* con la AWS regione in cui desideri creare il cluster.
4. Sostituisci *`my-cluster`* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
5. Sostituisci *`1.36`* con qualsiasi versione supportata da Amazon EKS.
6. Modifica i valori di `vpc-private-subnets` in base alle tue esigenze. Puoi inoltre aggiungere altri ID. Devi specificare almeno due ID sottorete, tuttavia, se preferisci specificare le sottoreti pubbliche, puoi modificare `--vpc-private-subnets` in `--vpc-public-subnets`. Alle sottoreti pubbliche è associata una tabella di routing con un routing a un gateway Internet, al contrario delle sottoreti private. Per questo motivo, ti consigliamo di utilizzare le sottoreti private quando possibile.

   Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.
7. Esegui il comando seguente:

   ```
   eksctl create cluster --name my-cluster --region region-code --version 1.36 --vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
   ```

   Il provisioning del cluster richiede diversi minuti. Durante la creazione del cluster, vengono visualizzate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

   ```
   [#]  EKS cluster "my-cluster" in "region-code" region is ready
   ```
8. Continua con the section called "Passaggio 3: aggiornamento di kubeconfig"

Impostazioni facoltative

Per visualizzare la maggior parte delle opzioni che è possibile specificare durante la creazione di un cluster con `eksctl`, utilizza il comando `eksctl create cluster --help`. Per visualizzare tutte le opzioni disponibili, puoi utilizzare un file `config`. Per ulteriori informazioni, consulta Uso dei file config e lo Schema dei file config nella documentazione di `eksctl`. Gli esempi di file di configurazione sono disponibili su GitHub.

Di seguito sono riportate le impostazioni facoltative da aggiungere al comando precedente, quando necessario. Puoi abilitare queste opzioni esclusivamente durante la creazione del cluster, non in seguito. Per specificare queste opzioni, non usare il comando precedente. Crea invece il cluster con un file di configurazione eksctl e specifica tali impostazioni.

- Se vuoi specificare uno o più gruppi di sicurezza assegnati da Amazon EKS alle interfacce di rete create, scegli l'opzione securityGroup.

  Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.
- Se desideri specificare da quale blocco CIDR (`IPv4`Classless Inter-domain Routing) Kubernetes assegna gli indirizzi IP del servizio, specifica l'opzione ServiceIPv4CIDR. https://eksctl.io/usage/schema/#kubernetesNetworkConfig-serviceIPv4CIDR

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

  - Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
  - Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
  - Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

  Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`. Per i cluster con nodi ibridi abilitati, Amazon EKS può selezionare un CIDR diverso per evitare conflitti con le reti di nodi remoti e pod configurati.
- Se stai creando un cluster e desideri che il cluster assegni `IPv6` indirizzi a pod e servizi anziché `IPv4` indirizzi, specifica l'opzione IPFamily.

Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti the section called "Considerazioni e requisiti relativi al VPC", the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

## Crea cluster - AWS console

1. Aprire la Console Amazon EKS.
2. Seleziona Aggiungi cluster e quindi Crea.
3. In Opzioni di configurazione, seleziona Configurazione personalizzata
   - Per informazioni sulla creazione rapida di un cluster con EKS Auto Mode, vederethe section called "Console di gestione".
4. In modalità automatica di EKS, disattiva l'opzione Usa modalità automatica di EKS.
   - Per informazioni sulla creazione di cluster della modalità automatica di EKS con configurazione personalizzata, consulta the section called "Creare un cluster automatico".
5. Nella pagina Configura cluster compila i campi seguenti:
   - Nome: un nome per il cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Ruolo IAM del cluster: scegli il ruolo IAM del cluster Amazon EKS che hai creato per consentire al piano di controllo Kubernetes di gestire AWS le risorse per tuo conto.
   - Versione Kubernetes – la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra una versione precedente.
   - Tipo di supporto: la policy della versione di Kubernetes che vuoi impostare per il tuo cluster. Se vuoi che il cluster venga eseguito solo su una versione di supporto standard, puoi scegliere Supporto standard. Se vuoi che il tuo cluster acceda al supporto esteso alla fine del supporto standard per una versione, puoi scegliere Supporto esteso. Se selezioni una versione di Kubernetes che attualmente è in supporto esteso, non puoi selezionare il supporto standard come opzione.

- Crittografia dei segreti (Facoltativo): scegli di abilitare la crittografia dei segreti per i segreti Kubernetes tramite una chiave KMS. Puoi abilitare questa funzionalità anche dopo la creazione del cluster. Assicurati, tuttavia, di acquisire familiarità con le informazioni contenute nella sezione the section called "Abilitare la crittografia dei segreti".
- Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Tagging delle risorse ".
- ARC Zonal shift — (Facoltativo) Puoi utilizzare Amazon Application Recovery Controller per mitigare le zone di disponibilità compromesse. Per ulteriori informazioni, consulta the section called "Ulteriori informazioni relative allo spostamento zonale".

6. Nella sezione Accesso al cluster della pagina Configura cluster, inserisci i seguenti campi:
   - Accesso amministratore cluster Bootstrap: il creatore del cluster diventa automaticamente amministratore Kubernetes. Se vuoi disabilitare questa opzione, seleziona Impedisci l'accesso all'amministratore del cluster.
   - Modalità di autenticazione del cluster: scegli come concedere l'accesso alle API Kubernetes agli utenti e ai ruoli IAM. Per ulteriori informazioni, consulta the section called "Set Cluster Authentication Mode".

   Quando hai finito con questa pagina, seleziona Avanti.
7. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:
   - VPC: crea il cluster in un VPC esistente conforme ai requisiti su VPC di Amazon EKS. Prima di scegliere un VPC, ti consigliamo di acquisire familiarità con i requisiti e le considerazioni nella sezione the section called "Requisiti del VPC e delle sottoreti". Dopo la creazione del cluster, non potrai più modificare il VPC da utilizzare. Se l'elenco risulta vuoto, è necessario crearne uno prima. Per ulteriori informazioni, consulta the section called "Crea un VPC".
   - Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. È necessario selezionarne almeno due.

     Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.

     Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS.

     Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo

gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.

- Scegli la famiglia di indirizzi IP del cluster: puoi scegliere IPv4 o IPv6.

  Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

- (Facoltativo) Scegli Configure Kubernetes Service IP address range (Configura l'intervallo di indirizzi IP del servizio Kubernetes) e specifica un intervallo **`IPv4`** del servizio.

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

  - Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
  - Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
  - Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`. Per i cluster con nodi ibridi abilitati, Amazon EKS può selezionare un CIDR diverso per evitare conflitti con le reti di nodi remoti e pod configurati.

- Per Cluster endpoint access (Accesso all'endpoint del cluster), seleziona un'opzione. Puoi modificare questa opzione dopo aver creato il cluster. Prima di selezionare un'opzione non predefinita, assicurati di familiarizzare con le opzioni e le relative implicazioni. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

- (Facoltativo) Per l'uscita dal piano di controllo, scegli come indirizzare il traffico in uscita dal piano di controllo al tuo VPC. L'impostazione predefinita è Amazon EKS managed. Se selezioni il cliente instradato, diventi responsabile del percorso di rete dal piano di controllo alle tue risorse VPC. Per ulteriori informazioni, consulta the section called "Uscita del piano di controllo".

  Quando hai finito con questa pagina, seleziona Avanti.

8. (Facoltativo) Nella pagina Configura osservabilità, scegli le opzioni Parametri e Registrazione del piano di controllo da attivare. Per impostazione predefinita, i tipi di log sono disattivati.
   - Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Passaggio 1: attivazione delle metriche Prometheus".
   - Per ulteriori informazioni sulle opzioni Registrazione del piano di controllo, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

   Quando hai finito con questa pagina, seleziona Avanti.
9. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster. Certi componenti aggiuntivi sono preselezionati. Puoi scegliere tutti i componenti aggiuntivi di Amazon EKS e AWS Marketplace di cui hai bisogno. Se i componenti aggiuntivi di AWS Marketplace che desideri installare non sono elencati, puoi scegliere la numerazione delle pagine per visualizzare ulteriori risultati di pagina o cercare i componenti aggiuntivi di AWS Marketplace disponibili inserendo del testo nella casella di ricerca. Puoi anche filtrare per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca. Quando crei un cluster, puoi visualizzare, selezionare e installare tutti i componenti aggiuntivi che supportano EKS Pod Identity, come descritto in dettaglio in the section called "Pod Identity".

   Quando hai finito con questa pagina, seleziona Avanti.

   Alcuni componenti aggiuntivi, come Amazon VPC CNI, CoreDNS e kube-proxy, sono installati per impostazione predefinita. Disabilitare uno dei componenti aggiuntivi predefiniti potrebbe influire sulla tua capacità di eseguire le applicazioni Kubernetes.
10. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona la versione da installare. Dopo la creazione del cluster, puoi eseguire in qualsiasi momento l'aggiornamento a una versione successiva.

    Per i componenti aggiuntivi che supportano EKS Pod Identities, puoi utilizzare la console per generare automaticamente il ruolo con il nome, la politica AWS gestita e la politica di fiducia precompilati specificamente per il componente aggiuntivo. È possibile riutilizzare i ruoli esistenti

o crearne di nuovi per i componenti aggiuntivi supportati Per i passaggi necessari per usare la console per creare ruoli per i componenti aggiuntivi che supportano EKS Pod Identity, consulta the section called "Crea componente aggiuntivo (AWS Console)". Se un componente aggiuntivo non supporta EKS Pod Identity, viene mostrato un messaggio con le istruzioni per usare la procedura guidata per creare i ruoli IAM per gli account di servizio (IRSA) dopo la creazione del cluster.

Puoi aggiornare la configurazione di ogni componente aggiuntivo dopo la creazione del cluster. Per ulteriori informazioni sulla configurazione dei componenti aggiuntivi, consulta the section called "Aggiornamento di un componente aggiuntivo". Quando hai finito con questa pagina, seleziona Avanti.

11. Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE.

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Il provisioning del cluster richiede diversi minuti.

12. Continua con the section called "Passaggio 3: aggiornamento di kubeconfig"

## Crea cluster - AWS CLI

1. Crea un cluster con il comando seguente. Prima di eseguire il comando, apporta le modifiche seguenti:
   - Sostituiscilo *region-code* con la AWS regione in cui desideri creare il cluster.
   - Sostituisci *my-cluster* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100

caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.

- Sostituisci *1.36* con qualsiasi versione supportata da Amazon EKS.
- Sostituisci *111122223333* con il tuo ID account e *myAmazonEKSClusterRole* con il nome del ruolo IAM del cluster.
- Sostituisci i valori di `subnetIds` con quelli in tuo possesso. Puoi inoltre aggiungere altri ID. Devi specificare almeno due ID sottorete,

  Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.
- Se non vuoi specificare un ID del gruppo di sicurezza, rimuovi `,securityGroupIds=sg-<ExampleID1>` dal comando. Se vuoi specificare uno o più ID del gruppo di sicurezza, sostituisci i valori di `securityGroupIds` con quelli in tuo possesso. Puoi inoltre aggiungere altri ID.

  Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.
- (Facoltativo) Per gestire tu stesso il percorso di uscita dal piano di controllo al tuo VPC, aggiungi `controlPlaneEgressMode=CUSTOMER_ROUTED` a. `--resources-vpc-config` Il valore predefinito è `AWS_MANAGED`. Puoi modificare questa impostazione anche dopo la creazione del cluster. Per ulteriori informazioni, consulta the section called "Uscita del piano di controllo".

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-version
 1.36 \
   --role-arn arn:aws: iam::111122223333:role/myAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=subnet-ExampleID1,subnet-
ExampleID2,securityGroupIds=sg-ExampleID1
```

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In

questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Di seguito sono riportate le impostazioni facoltative da aggiungere al comando precedente, quando necessario. Puoi abilitare queste opzioni esclusivamente durante la creazione del cluster, non in seguito.

- Durante la creazione del cluster, EKS installa più componenti aggiuntivi di rete per impostazione predefinita. Sono inclusi Amazon VPC CNI, CoreDNS e kube-proxy.

  Se vuoi disabilitare l'installazione di questi componenti aggiuntivi di rete predefiniti, usa il parametro seguente. Può essere usato per CNI alternativi, come Cilium. Per ulteriori informazioni, consulta la Documentazione di riferimento delle API di EKS.

  ```
  aws eks create-cluster --no-bootstrap-self-managed-addons …
  ```

- Se desideri specificare da quale blocco `IPv4` Classless Inter-domain Routing (CIDR) Kubernetes assegna gli indirizzi IP del servizio, devi specificarlo aggiungendo il comando al seguente comando. `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>`

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

  - Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
  - Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
  - Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`. Per i cluster con nodi ibridi abilitati, Amazon EKS può selezionare un CIDR diverso per evitare conflitti con le reti di nodi remoti e pod configurati.

- Se stai creando un cluster e vuoi che il cluster assegni gli indirizzi IPv6 ai pod e ai servizi anziché a indirizzi IPv4, aggiungi `--kubernetes-network-config ipFamily=ipv6` al seguente comando.

  Kubernetes assegna indirizzi IPv4 ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia IPv6, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia IPv6, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio IPv6 come puoi fare per la famiglia IPv4. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

2. Il provisioning del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

   Non passare alla fase successiva finché l'output restituito non è ACTIVE.

3. Continua con the section called "Passaggio 3: aggiornamento di kubeconfig"

## Passaggio 3: aggiornamento di kubeconfig

1. Se hai creato il cluster utilizzando `eksctl`, puoi ignorare questo passaggio, poiché è già stato completato da `eksctl`. Abilita `kubectl` per consentire la comunicazione con il cluster aggiungendo un nuovo contesto al file `config` `kubectl`. Per ulteriori informazioni su come creare e aggiornare il file, consulta the section called "Accedere al cluster con kubectl".

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

   Di seguito viene riportato un output di esempio:

```
Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

2. Conferma la comunicazione con il cluster eseguendo il comando seguente.

```
kubectl get svc
```

Di seguito viene riportato un output di esempio.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

## Passaggio 4: configurazione del cluster

1. (Consigliato) Per utilizzare alcuni componenti aggiuntivi di Amazon EKS o per consentire ai singoli carichi di lavoro Kubernetes di avere autorizzazioni IAM ( AWS Identity and Access Management) specifiche, crea un provider IAM OpenID Connect (OIDC) per il tuo cluster. È necessario creare un provider OIDC IAM per il cluster una sola volta. Per ulteriori informazioni sui componenti aggiuntivi di Amazon EKS, consulta the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'assegnazione di autorizzazioni IAM specifiche ai carichi di lavoro, consulta the section called "Credenziali con IRSA".
2. (Consigliato) Configura il cluster per il componente aggiuntivo CNI di Amazon VPC per Kubernetes prima di implementare i nodi Amazon EC2 nel cluster. Per impostazione predefinita, il plugin è stato installato con il cluster. Quando aggiungi i nodi Amazon EC2 al cluster, il plugin viene implementato automaticamente su ogni nodo Amazon EC2 aggiunto. Il componente aggiuntivo richiede di collegare una delle policy IAM seguenti a un ruolo IAM. Se il cluster usa la famiglia `IPv4`, usa la policy IAM gestita AmazonEKS_CNI_Policy. Se il tuo cluster usa la famiglia `IPv6`, utilizza una policy IAM creata da te.

   Il ruolo IAM a cui si collega la policy può essere il ruolo IAM del nodo o un ruolo dedicato utilizzato solo per il plugin. Consigliamo di allegare la policy a quest'ultimo ruolo. Per ulteriori informazioni sulla creazione del ruolo, consulta the section called "Configurare per IRSA" o il ruolo IAM del nodo Amazon EKS.
3. Se hai distribuito il cluster utilizzando il, puoi saltare questo passaggio. Console di gestione AWS Per impostazione predefinita, Console di gestione AWS distribuisce il plug-in Amazon VPC CNI per i componenti aggiuntivi Kubernetes, CoreDNS e Amazon EKS. `kube-proxy`

   Se distribuisci il cluster utilizzando uno `eksctl` o l'interfaccia a riga di comando, vengono distribuiti il plug-in Amazon VPC AWS CNI per Kubernetes, CoreDNS e i componenti aggiuntivi autogestiti. `kube-proxy` Puoi eseguire la migrazione dei componenti aggiuntivi CNI di Amazon

VPC per Kubernetes, CoreDNS e dei componenti autogestiti `kube-proxy` implementati con il cluster verso i componenti aggiuntivi di Amazon EKS. Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi Amazon EKS".

4. (Facoltativo) Se non l'hai già fatto, puoi abilitare le metriche Prometheus per il cluster. Per ulteriori informazioni, consulta Creare uno scraper nella Guida per l'utente di Amazon Managed Service for Prometheus.
5. Se hai intenzione di implementare nel tuo cluster carichi di lavoro che utilizzano volumi Amazon EBS, devi installare il Amazon EBS CSI nel cluster prima di implementare i carichi di lavoro.

## Fasi successive

- L'utente o il principale IAM che ha creato il cluster è l'unico principale in grado di accedere al cluster. Concedi autorizzazioni ad altri principali IAM in modo che possano accedere al tuo cluster.
- Se il principale IAM che ha creato il cluster dispone solo delle autorizzazioni IAM minime a cui si fa riferimento nei prerequisiti, potresti voler aggiungere ulteriori autorizzazioni Amazon EKS per quel principale IAM. Per ulteriori informazioni su come concedere le autorizzazioni Amazon EKS ali principali IAM, consulta the section called "Riferimento IAM".
- Se desideri che il principale IAM che ha creato il cluster o qualsiasi altro principale visualizzi le risorse Kubernetes nella console Amazon EKS, concedi le autorizzazioni necessarie alle entità.
- Se desideri che i nodi e i principali IAM accedano al cluster dall'interno della VPC, attiva l'endpoint privato per il cluster. L'endpoint pubblico è abilitato per impostazione predefinita. Puoi disabilitare l'endpoint pubblico dopo aver abilitato l'endpoint privato, se lo desideri. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".
- Abilitare la crittografia dei segreti per il cluster.
- Configurare la registrazione per il cluster.
- Aggiungi nodi al cluster.

# Piano di controllo predisposto di Amazon EKS

## Panoramica di

Amazon EKS Provisioned Control Plane è una funzionalità che consente agli amministratori del cluster di scegliere tra una serie di livelli di scalabilità e di designare il livello prescelto per ottenere prestazioni molto elevate e prevedibili dal piano di controllo del cluster. Ciò consente agli

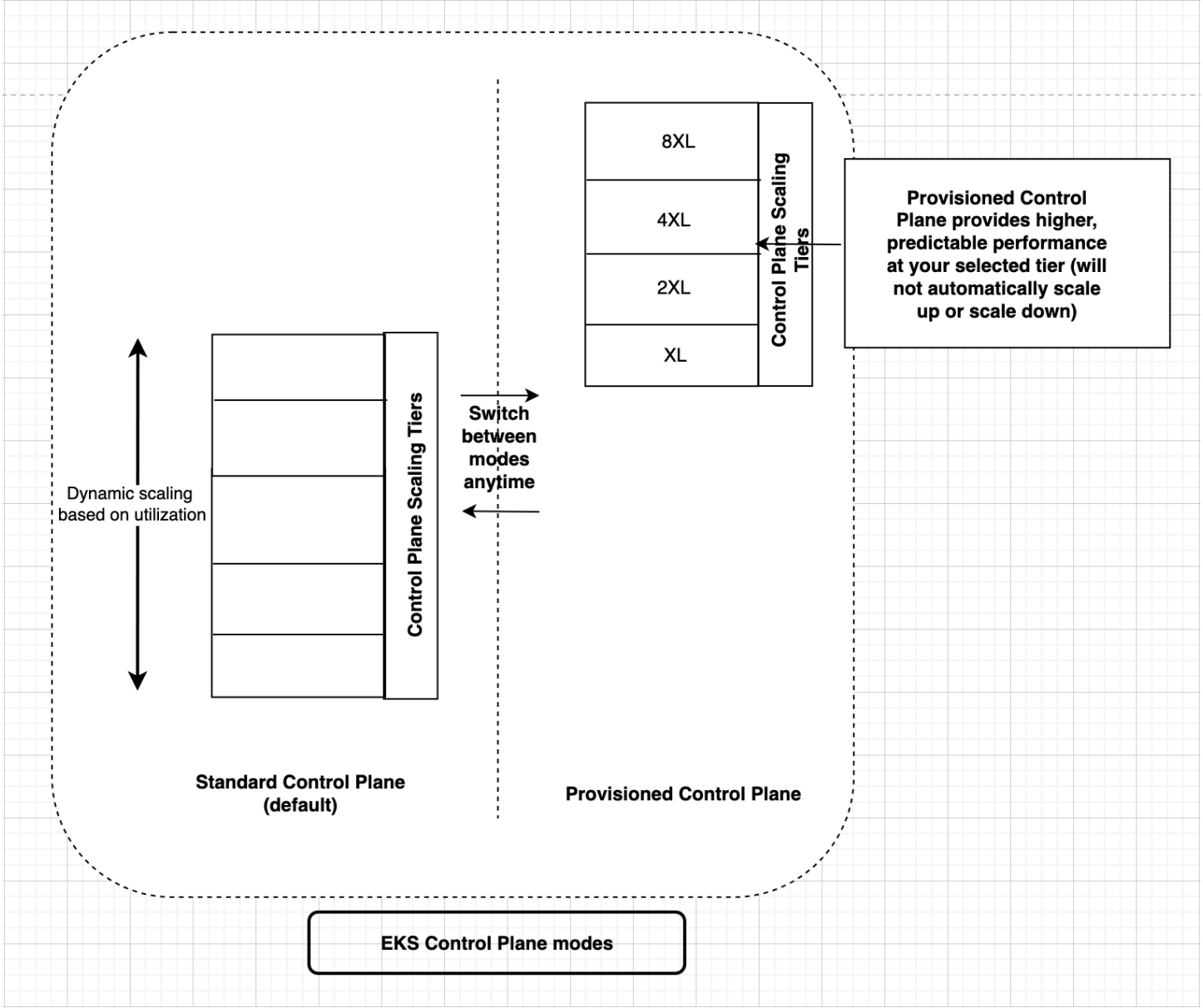
amministratori del cluster di garantire che il piano di controllo sia sempre dotato della capacità specificata.

Amazon EKS offre due modalità operative per il piano di controllo del cluster. Per impostazione predefinita, i cluster Amazon EKS utilizzano la modalità  Standard, in cui il piano di controllo aumenta e diminuisce automaticamente in base alle esigenze del carico di lavoro. La modalità standard alloca dinamicamente una capacità del piano di controllo sufficiente per soddisfare le esigenze dei carichi di lavoro ed è la soluzione consigliata per la maggior parte dei casi d'uso. Tuttavia, per carichi di lavoro specializzati che non possono tollerare alcuna variabilità delle prestazioni dovuta alla scalabilità del piano di controllo o che richiedono quantità molto elevate di capacità del piano di controllo, è possibile utilizzare facoltativamente la modalità Provisioned.  La modalità provisioned consente di preallocare la capacità del piano di controllo, che è sempre pronta a gestire requisiti di carico di lavoro impegnativi.

> **Note**
>
> La modalità Provisioned è una modalità operativa aggiuntiva del piano di controllo oltre alla modalità Standard predefinita. L'introduzione della modalità Provisioned non modifica il comportamento della modalità Standard.

Con EKS Provisioned Control Plane, gli amministratori del cluster possono predisporre in anticipo la capacità del piano di controllo desiderata, fornendo prestazioni prevedibili e elevate dal piano di controllo del cluster che è sempre disponibile. EKS Provisioned Control Plane consente inoltre agli amministratori di cluster di fornire la stessa capacità del piano di controllo in tutti gli ambienti, dallo staging alla produzione e ai siti di disaster recovery. Ciò è importante per garantire che le prestazioni del piano di controllo ottenute in tutti gli ambienti siano coerenti e prevedibili. Infine, EKS Provisioned Control Plane consente di accedere a livelli molto elevati di prestazioni del piano di controllo, consentendo l'esecuzione di carichi di lavoro di intelligenza artificiale estremamente scalabili, calcolo ad alte prestazioni e carichi di lavoro di elaborazione dati su larga scala su Kubernetes.

Tutti i cluster Amazon EKS esistenti e nuovi funzionano in modalità Standard per impostazione predefinita. Per i cluster che richiedono prestazioni elevate e prevedibili dal piano di controllo, puoi scegliere di utilizzare la funzione EKS Provisioned Control Plane. Ti verrà addebitata la tariffa oraria per il particolare livello di scalabilità del piano di controllo, oltre alle tariffe orarie EKS del supporto standard o esteso. Per ulteriori informazioni sui prezzi, consulta i prezzi di  Amazon EKS.




## Casi d'uso

EKS Provisioned Control Plane è progettato per affrontare scenari specifici in cui prestazioni elevate e prevedibili del piano di controllo sono fondamentali per le tue operazioni. La comprensione di questi casi d'uso può aiutarti a determinare se EKS Provisioned Control Plane è la soluzione giusta per i tuoi carichi di lavoro.

Performance-critical carichi di lavoro — Per i carichi di lavoro che richiedono una latenza minima e le massime prestazioni dal piano di controllo di Kubernetes, EKS Provisioned Control Plane offre una capacità che elimina la variabilità delle prestazioni grazie alla scalabilità del piano di controllo.

Carichi di lavoro altamente scalabili: se esegui carichi di lavoro altamente scalabili come l'addestramento e l'inferenza AI, il calcolo ad alte prestazioni o l'elaborazione di dati su larga scala che richiedono un numero elevato di nodi in esecuzione nel cluster, Provisioned Control Plane fornisce la capacità del piano di controllo necessaria per supportare questi carichi di lavoro impegnativi.

Eventi previsti ad alta richiesta: quando si prevede un improvviso aumento delle richieste dei piani di controllo a causa di un evento imminente, come vendite o promozioni di e-commerce, lanci di prodotti, festività natalizie o importanti eventi sportivi o di intrattenimento, Provisioned Control Plane consente di scalare in anticipo la capacità del piano di controllo. Questo approccio proattivo assicura che il piano di controllo sia pronto a gestire l'aumento del carico senza attendere che il ridimensionamento automatico risponda alla domanda.

Mission-critical alta disponibilità - Per i carichi di lavoro che richiedono i massimi livelli di disponibilità, Provisioned Control Plane offre un Service Level Agreement (SLA) del 99,99%, misurato a intervalli di 1 minuto. Questo SLA migliorato prevede un impegno di disponibilità più rigoroso rispetto allo SLA del 99,95% della modalità Standard Control Plane, misurato a intervalli di 5 minuti.

Coerenza dell'ambiente: il Provisioned Control Plane consente di abbinare la capacità e le prestazioni del piano di controllo in tutti gli ambienti di staging e produzione, aiutando a identificare potenziali problemi in anticipo prima dell'implementazione in produzione. Mantenendo lo stesso livello del piano di controllo in tutti gli ambienti, è possibile garantire che i risultati dei test riflettano accuratamente il comportamento di produzione, riducendo il rischio di sorprese legate alle prestazioni durante l'implementazione.

Disaster recovery e business continuity: per gli scenari di disaster recovery, Provisioned Control Plane consente di effettuare il provisioning degli ambienti di failover con lo stesso livello di capacità dell'ambiente principale. Ciò garantisce interruzioni minime e un ripristino rapido durante gli eventi di failover, poiché il cluster di disaster recovery avrà caratteristiche prestazionali del piano di controllo identiche a quelle del cluster di produzione dal momento in cui viene attivato.

## Livelli di scalabilità del piano di controllo

EKS Provisioned Control Plane offre livelli di scalabilità denominati in base alle taglie delle magliette (XL, 2XL, 4XL e 8XL). Ogni livello definisce la propria capacità attraverso quattro attributi chiave che determinano le caratteristiche prestazionali del piano di controllo del cluster. La comprensione di questi attributi ti aiuta a selezionare il livello appropriato per i requisiti del carico di lavoro.

La concorrenza delle richieste API  misura il numero di richieste che il server API del piano di controllo Kubernetes può elaborare contemporaneamente, il che è fondamentale per carichi di lavoro ad alto rendimento.

La frequenza di pianificazione dei pod  indica la velocità con cui lo scheduler Kubernetes predefinito può pianificare i pod sui nodi, misurata in pod al secondo.

La dimensione del database del cluster  indica lo spazio di archiviazione allocato a etcd, il database che contiene il cluster. state/metadata

Lo SLA (Service Level Agreement)  indica l'impegno mensile di uptime dell'endpoint Kubernetes del cluster EKS

Quando si esegue il provisioning del piano di controllo del cluster su un determinato livello di scalabilità utilizzando Provisioned Control Plane, EKS assicura che il piano di controllo del cluster mantenga i limiti corrispondenti a quel livello. I limiti dei livelli di scalabilità del piano di controllo variano in base alla versione di Kubernetes, come mostrato nelle tabelle seguenti.

## Da EKS da v1.30 a v1.33

| Livello di scalabilità del piano di controllo fornito | Concorrenza delle richieste API (postazioni) | Tariffa di pianificazione dei pod () pods/sec | Dimensione del database del cluster (GB) | SLA (misurato a intervalli di 1 minuto) |
|---|---|---|---|---|
| XL | 1700 | 167 | 16 | 99,99% |
| 2XL | 3400 | 283 | 16 | 99,99% |
| 4XL | 6800 | 400 | 16 | 99,99% |
| 8XL | 13600 | 400 | 16 | 99,99% |

## EKS v1.34 e versioni successive

| Livello di scalabilità del piano di controllo fornito | Concorrenza delle richieste API (postazioni) | Tariffa di pianificazione dei pod () pods/sec | Dimensione del database del cluster (GB) | SLA (misurato a intervalli di 1 minuto) |
|---|---|---|---|---|
| XL | 2000 | 167 | 16 | 99,99% |
| 2XL | 4000 | 283 | 16 | 99,99% |
| 4XL | 8000 | 400 | 16 | 99,99% |
| 8XL | 16000 | 400 | 16 | 99,99% |

### Recupero programmatico dei valori dei livelli di scalabilità

Puoi utilizzare l'`DescribeClusterVersions`API per recuperare gli attributi del livello di scalabilità e i relativi valori per ogni versione di Kubernetes. Ciò è utile quando si automatizza la selezione dei livelli o è necessario confrontare le funzionalità dei livelli tra le versioni a livello di programmazione. Per ulteriori informazioni, consulta  Amazon EKS  DescribeClusterVersions  API Reference.

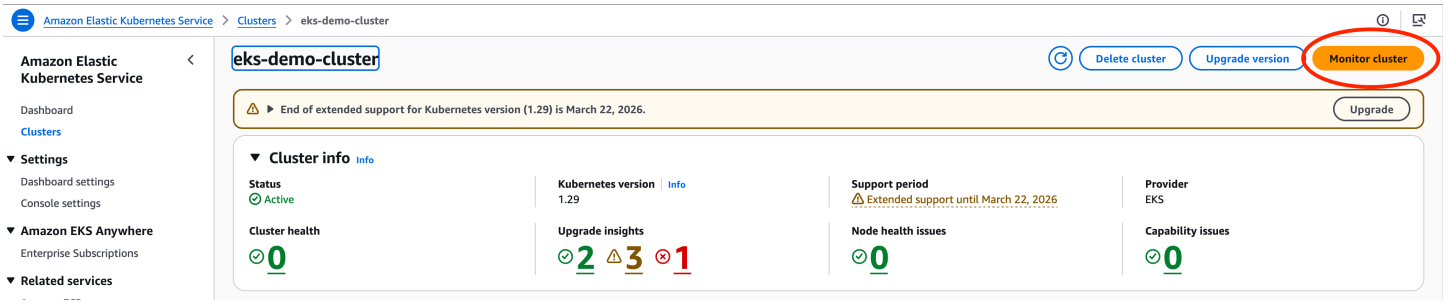
### Monitoraggio del piano di controllo, scalabilità e utilizzo dei livelli

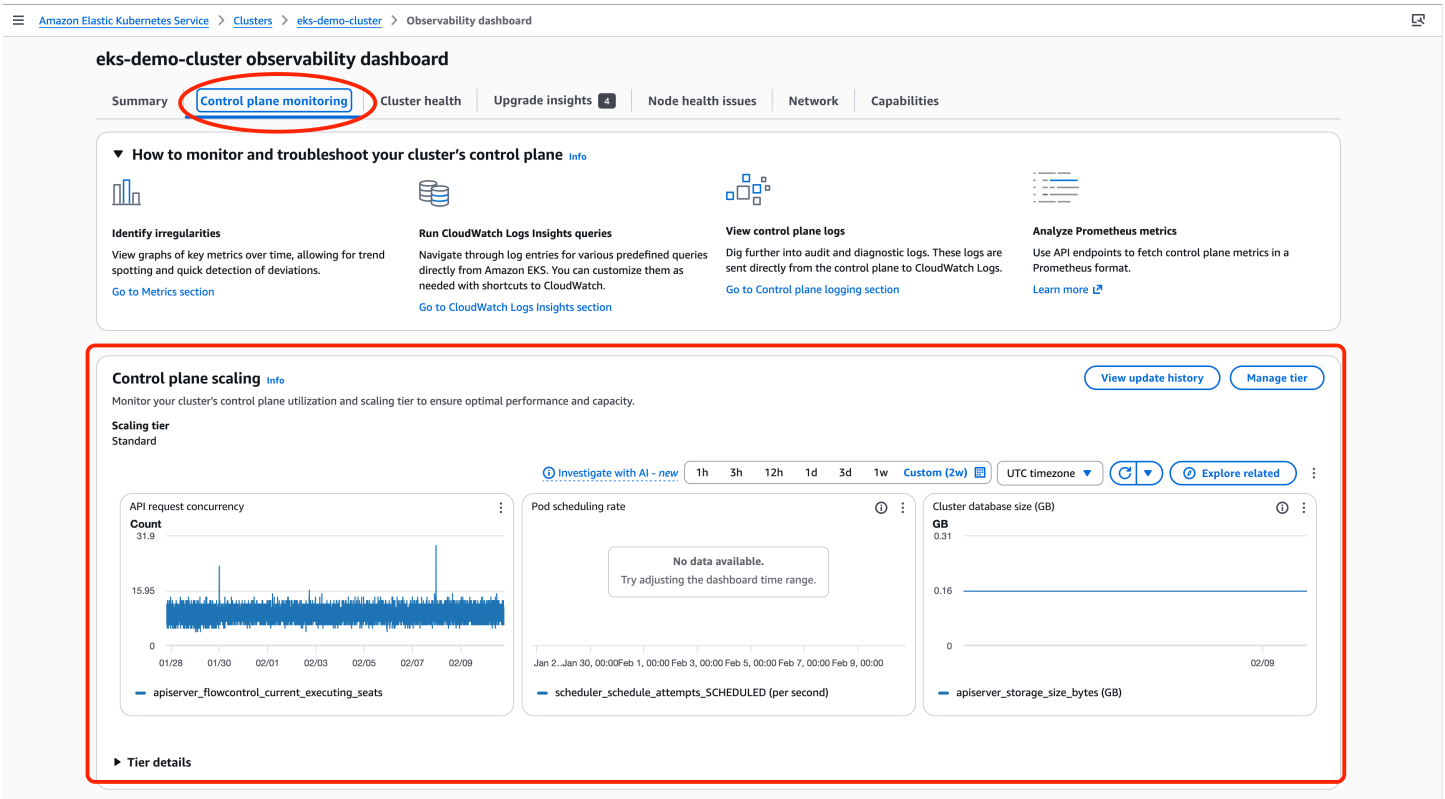
Amazon EKS fornisce diverse metriche per aiutarti a monitorare l'utilizzo dei livelli del tuo piano di controllo. Queste metriche sono pubblicate come parametri  Amazon  e sono accessibili tramite la CloudWatch console e CloudWatch EKS. Inoltre, queste metriche sono scaricabili dall'endpoint Prometheus del cluster EKS (vedi qui).  ???

| | Metrica Prometheus | CloudWatch Parametro |
|---|---|---|
| Concorrenza delle richieste API | apiserver_flowcontrol_current_executing_seats | apiserver_flowcontrol_current_executing_seats |
| Frequenza di pianificazione dei pod | scheduler_schedule_attempts_total | scheduler_schedule_attempts_total, Scheduler_schedule_attempts_Scheduled, |

| | Metrica Prometheus | CloudWatch Parametro |
|---|---|---|
| | | Scheduler_schedule_attempts_unschedulable |
| Dimensioni del database del cluster | apiserver_storage_size_bytes (NOTA: etcd_mvcc_db_total_size_in_use_in_bytes verrà implementato come metrica prometheus per tutti i cluster EKS entro la seconda metà del 2026) | etcd_mvcc_db_total_size_in_use_in_bytes |

Puoi visualizzare l'utilizzo del piano di controllo nella console Amazon EKS. Dalla pagina di panoramica del cluster, scegli Monitor cluster per accedere al dashboard di osservabilità, quindi seleziona la scheda Monitoraggio del piano di controllo per visualizzare l'utilizzo del piano di controllo nella sezione Ridimensionamento del piano di controllo.






## Comprensione della capacità dei livelli rispetto alle prestazioni effettive

Quando si seleziona un livello di scalabilità del Provisioned Control Plane, gli attributi del livello rappresentano le configurazioni sottostanti che Amazon EKS applica al piano di controllo. Tuttavia, le prestazioni effettive ottenute dipendono dai modelli di carico di lavoro specifici, dalle configurazioni e dall'aderenza alle best practice di Kubernetes. Ad esempio, mentre un livello 4XL configura API Priority and Fairness (APF) con 6.800 postazioni di richieste simultanee, il throughput effettivo delle richieste ottenuto dal piano di controllo dipende dal tipo di operazioni eseguite. Ad esempio, Kubernetes penalizza le richieste di lista più di quelle get, e quindi il numero effettivo di richieste di lista elaborate contemporaneamente dal piano di controllo è inferiore alle richieste get. Per ulteriori informazioni, consulta la sezione API Priority and Fairness nella EKS Best Practices Guide. Allo stesso modo, sebbene lo scheduler predefinito QPS sia impostato su 400 per un livello 4XL, la velocità effettiva di pianificazione dei pod dipende da fattori come la disponibilità e l'integrità dei nodi per la pianificazione. Per ottenere prestazioni ottimali, assicurati che le tue applicazioni seguano le best practice di Kubernetes e siano configurate correttamente per le caratteristiche del carico di lavoro. Per ulteriori informazioni, consulta le best practice sulla scalabilità nella EKS Best Practices Guide.

## Concorrenza di sincronizzazione Horizontal Pod Autoscaler

Oltre agli attributi del livello Control Plane Scaling, i cluster EKS Provisioned Control Plane sono ottimizzati con una maggiore concorrenza di sincronizzazione Horizontal Pod Autoscaler (HPA). La concorrenza di sincronizzazione HPA è il numero di oggetti HPA che il controller manager Kubernetes elabora in parallelo. L'impostazione predefinita upstream di Kubernetes è 5. Una maggiore concorrenza consente al piano di controllo di riconciliare più oggetti HPA contemporaneamente, riducendo il tempo che intercorre tra una modifica del carico e la corrispondente azione di ridimensionamento. Ciò è utile per i cluster che eseguono un numero elevato di oggetti HPA, dove l'elaborazione in serie può ritardare la scalabilità automatica.

Ogni livello di scalabilità del Provisioned Control Plane è configurato con la seguente concorrenza di sincronizzazione HPA.

| Livello di scalabilità del Control Plane con provisioning | Concorrenza di sincronizzazione HPA |
|---|---|
| XL | 50 |
| 2XL | 100 |
| 4XL | 200 |
| 8XL | 200 |

È possibile osservare l'effetto di una maggiore concorrenza di sincronizzazione HPA tramite la metrica della coda di lavoro di Kubernetes controller manager per il controller HPA:

- `workqueue_depth{name="horizontalpodautoscaler"}`— il numero di oggetti HPA in attesa di essere elaborati dal controller HPA. Con una maggiore concorrenza, più lavoratori esauriscono questa coda in parallelo, quindi la profondità rimane bassa e stabile (idealmente raggiungendo lo zero in ogni periodo di riconciliazione). Una profondità della coda di lavoro pari o prossima allo zero indica che il piano di controllo è al passo con gli oggetti HPA del cluster e che le decisioni relative alla scalabilità automatica non sono ritardate da un arretrato.

L'effettiva reattività di scalabilità automatica osservata dipende anche da fattori esterni al piano di controllo. Ogni riconciliazione HPA recupera le metriche che determinano la decisione di scalabilità, in genere dal Kubernetes Metrics Server per l'utilizzo della CPU e della memoria o da un adattatore

di metriche personalizzato o esterno. Il numero di oggetti HPA nel cluster e la latenza di queste ricerche metriche influiscono entrambi sulla velocità con cui il controller può elaborare la coda. Se la fonte delle metriche non è in grado di soddisfare le richieste con sufficiente rapidità, le riconciliazioni rallentano indipendentemente dal numero di richieste elaborate in parallelo. L'esecuzione di un numero sufficiente di repliche di Metrics Server per soddisfare il volume delle richieste e l'ambito delle query metriche in modo efficiente ti aiuta a sfruttare appieno la maggiore concorrenza di sincronizzazione HPA. Per ulteriori informazioni su come funziona Horizontal Pod Autoscaler e su dove recupera le metriche, vedi Horizontal Pod Autoscaling sul sito web di Kubernetes. https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/

## Considerazioni

- Capacità del piano di controllo standard — EKS La modalità piano di controllo standard offre il miglior rapporto prezzo/prestazioni ed è l'opzione consigliata per la maggior parte dei casi d'uso. Tuttavia, per carichi di lavoro specializzati che non possono tollerare alcuna variabilità delle prestazioni dovuta alla scalabilità del piano di controllo o che richiedono quantità molto elevate di capacità del piano di controllo, puoi opzionalmente prendere in considerazione l'utilizzo della modalità Provisioned.
- Opt-in obbligatorio: i cluster esistenti non passeranno automaticamente dal piano di controllo standard a un livello EKS Provisioned Control Plane, più costoso. È necessario attivare esplicitamente uno dei nuovi livelli di scalabilità EKS Provisioned Control Plane.
- Restrizione all'uscita: la modalità Standard Control Plane supporta fino a 8 GB di dimensioni del database cluster (etcd). Se la dimensione del database del cluster supera gli 8 GB mentre si utilizza la modalità Provisioned, non è possibile tornare alla modalità Standard finché non si riducono le dimensioni del database a meno di 8 GB. Ad esempio, se utilizzi 14 GB di spazio di archiviazione del database in modalità Provisioned, devi prima ridurre l'utilizzo del database a meno di 8 GB prima di tornare alla modalità Standard.
- Nessun ridimensionamento automatico dei livelli: EKS Provisioned Control Plane non esegue automaticamente il ridimensionamento tra i livelli. Una volta selezionato un livello di scalabilità, il piano di controllo del cluster rimane bloccato su quel livello, garantendo prestazioni coerenti e prevedibili. Tuttavia, hai la flessibilità di implementare la tua soluzione di scalabilità automatica monitorando le metriche di utilizzo dei livelli e utilizzando le API EKS Provisioned Control Plane per aumentare o diminuire quando queste metriche superano le soglie definite, offrendoti il pieno controllo sulla tua strategia di scalabilità e sull'ottimizzazione dei costi.
- Visualizzazione del livello corrente: puoi utilizzare la console Amazon EKS, l'interfaccia a riga di comando di Amazon Web Services o l'API per visualizzare l'attuale livello di scalabilità del piano

di controllo. Nella CLI, puoi eseguire il comando: `describe-cluster` `aws eks describe-cluster --name cluster-name`

- Tempo di transizione dei livelli: puoi utilizzare la console Amazon EKS, le API Amazon EKS o la CLI per uscire o passare da un livello di scalabilità all'altro. Amazon EKS ha introdotto un nuovo tipo di aggiornamento del cluster chiamato`ScalingTierConfigUpdate`, che puoi controllare per monitorare l'avanzamento della transizione. Dopo aver eseguito un comando di modifica del livello, puoi elencare gli aggiornamenti sul cluster per visualizzare un nuovo aggiornamento di tipo `ScalingTierConfigUpdate` con stato`Updating`. Lo stato cambia `Successful` al termine dell'aggiornamento o in `Failed` caso di errore. Il campo di errore nell'aggiornamento indica il motivo dell'errore. Non ci sono restrizioni sulla frequenza con cui è possibile passare da un livello all'altro. Il completamento della modifica del livello del piano di controllo richiede diversi minuti. Non ci sono tempi di inattività del server API durante questo processo, poiché EKS attiva nuovi server API prima di chiudere quelli vecchi.
- Selezione del livello ottimale: per determinare il livello di scalabilità ottimale del Provisioned Control Plane per il cluster, puoi eseguire un test di carico eseguendo il provisioning del cluster sul livello più alto (8XL). Quindi esegui un test di carico per simulare i picchi di domanda sul piano di controllo del cluster. Osserva le metriche di utilizzo dei livelli del piano di controllo al picco di carico e utilizza queste osservazioni come fattore guida per selezionare il livello appropriato per la modalità Provisioned.
- Prezzi del Provisioned Control Plane: ti verrà addebitata la tariffa oraria per il livello di scalabilità del Provisioned Control Plane in cui si trova il cluster. Ciò si aggiunge alle tariffe orarie del supporto standard o esteso. Per ulteriori informazioni, consulta la pagina dei prezzi di Amazon EKS.
- Livello di scalabilità più ampio: se intendi eseguire il cluster su un livello di scalabilità superiore a 8XL, contatta il team del tuo account Amazon Web Services per ulteriori informazioni sui prezzi.
- Supporto per versioni e regioni di Kubernetes: EKS Provisioned Control Plane è supportato in tutte le regioni commerciali e cinesi di Amazon Web Services. GovCloud Provisioned Control Plane funziona su EKS v1.28 e versioni successive.
- SLA più elevato - EKS Provisioned Control Plane offre un EKS SLA più elevato rispetto al Standard Control Plane. Scopri di più nella pagina EKS SLA.

## Guida introduttiva all'Amazon EKS Provisioned Control Plane

Questa guida illustra i passaggi per configurare e utilizzare EKS Provisioned Control Plane utilizzando AWS Console di gestione AWS CLI e.

## Prerequisiti

Prima di iniziare, assicurati di disporre dei seguenti elementi:

- AWS CLI: uno strumento da riga di comando per lavorare con AWS servizi, incluso Amazon EKS. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando. Dopo aver installato la AWS CLI, ti consigliamo di configurarla anche. Per ulteriori informazioni, consulta Configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. Nota che la AWS CLI v2 è necessaria per utilizzare l'opzione update-kubeconfig mostrata in questa pagina.
- Autorizzazioni IAM richieste: il responsabile della sicurezza IAM che stai utilizzando deve disporre delle autorizzazioni per lavorare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi AWS CloudFormation, un VPC e le risorse correlate. Per ulteriori informazioni, consulta Azioni e utilizzo dei ruoli collegati ai servizi nella Guida per l'utente IAM. È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:

```
aws sts get-caller-identity
```

> **Note**
>
> Ti consigliamo di completare la procedura descritta in questo argomento in una shell Bash. In alternativa, puoi apportare alcune modifiche alla tua shell per alcuni comandi di script, come i caratteri di continuazione della riga, e per il modo in cui le variabili vengono impostate e utilizzate. Inoltre, le regole di escape e di utilizzo delle virgolette per la shell (interprete di comandi) potrebbero essere diverse. Per ulteriori informazioni, consulta Uso delle virgolette con le stringhe nella AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

## Piano di controllo fornito da EKS - AWS CLI

Crea cluster con EKS Provisioned Control Plane Scaling Tier

```
aws eks create-cluster --name prod-cluster \
--role-arn arn:aws:iam::012345678910:role/eks-service-role-AWSServiceRoleForAmazonEKS-
J7ONKE3BQ4PI \
--resources-vpc-config subnetIds=subnet-6782e71e,subnet-
e7e761ac,securityGroupIds=sg-6979fe18 \
```

```
--control-plane-scaling-config tier=tier-xl
```

Risposta:


```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "controlPlaneScalingConfig": {
            "tier": "tier-xl"
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
            "clusterLogging": [
```

```
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "CREATING",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "health": {
            "issues": []
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
```

```
        }
    }
}
```

Visualizza il livello Control Plane Scaling del cluster

```
aws eks describe-cluster --name prod-cluster
```

Risposta:

```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "controlPlaneScalingConfig": {
            "tier": "tier-xl"
        },
```

```
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "ACTIVE",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
```

```
        },
        "health": {
            "issues": []
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
}
```

Aggiorna il cluster per utilizzare EKS Provisioned Control Plane

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=tier-2xl
```

Risposta:

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "ScalingTierConfigUpdate",
        "params": [
            {
                "type": "UpdatedTier",
                "value": "tier-2xl"
            },
            {
                "type": "PreviousTier",
                "value": "tier-xl"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

Visualizza l'aggiornamento di Control Plane Scaling

```
aws eks list-updates --name example
```

Risposta:

```
{
    "updateIds": [
        "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd1"
    ]
}
```

Uscire da Provisioned Control Plane a Standard Control Plane

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=standard
```

Risposta:

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "ScalingTierConfigUpdate",
        "params": [
            {
                "type": "UpdatedTier",
                "value": "standard"
            },
            {
                "type": "PreviousTier",
                "value": "tier-2xl"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

## Piano di controllo predisposto EKS - Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Crea cluster.
3. In Opzioni di configurazione, seleziona Configurazione personalizzata.
4. Scorri verso il basso fino al livello di scalabilità del piano di controllo. Seleziona Usa un livello di scalabilità per abilitare Provisioned Control Plane.

5. Seleziona il livello di scalabilità del piano di controllo che desideri fornire per il cluster tra varie opzioni di scaling tier come XL, 2XL, 4XL e 8XL.
6. Seleziona altre opzioni di configurazione del cluster in base alle esigenze. Nel passaggio finale seleziona Crea cluster. Nota che il completamento della creazione del cluster potrebbe richiedere alcuni minuti.

# Configurazione avanzata del piano di controllo Kubernetes

## Panoramica di

Amazon EKS gestisce il piano di controllo Kubernetes per il tuo cluster, inclusi il server API, lo scheduler e il controller manager. EKS esegue questi componenti con impostazioni Kubernetes upstream predefinite che funzionano bene per la maggior parte dei carichi di lavoro e non è necessario modificarle per la maggior parte dei cluster. Tuttavia, alcuni carichi di lavoro traggono vantaggio da diverse impostazioni del piano di controllo. Potresti volere che lo scheduler raggruppi i pod su un numero inferiore di nodi per ridurre i costi di elaborazione, conservare gli eventi Kubernetes per un periodo più breve per limitare la crescita dei database cluster (ecc.) o valutare più frequentemente le decisioni relative alla scalabilità automatica.

Con la configurazione avanzata del piano di controllo di Kubernetes, puoi impostare questi parametri direttamente sul tuo cluster. EKS li applica al piano di controllo e il cluster continua a funzionare con le stesse caratteristiche di disponibilità e prestazioni.

Si tratta di configurazioni avanzate. Ogni parametro di configurazione modifica il comportamento di un componente principale del piano di controllo di Kubernetes per i carichi di lavoro in esecuzione sul cluster e il valore giusto dipende dal carico di lavoro. Prima di modificare un parametro, leggi le considerazioni relative alle sezioni seguenti e verifica la modifica in un cluster non di produzione.

È possibile impostare i parametri di configurazione avanzati del piano di controllo quando si crea un cluster o aggiornarli su un cluster esistente in qualsiasi momento. Questa funzionalità utilizza le `UpdateClusterConfig` operazioni `CreateCluster` e le operazioni esistenti con nuovi parametri, quindi puoi impostarla tramite AWS CLI, AWS SDK o. Console di gestione AWS AWS CloudFormation EKS convalida ogni configurazione prima di applicarla e registra le modifiche. AWS CloudTrail

I parametri avanzati del piano di controllo si applicano all'intero cluster e a tutti i carichi di lavoro in esecuzione su di esso. Non è possibile applicarli a singoli namespace o carichi di lavoro. EKS

vincola ogni parametro a un intervallo convalidato. I valori supportati per ogni parametro sono elencati insieme a tale parametro nella sezione seguente.

## Parametri del piano di controllo Kubernetes supportati

Amazon EKS supporta i seguenti parametri. Ogni parametro appartiene a un componente del piano di controllo e viene impostato tramite il campo di configurazione relativo a quel componente:`kubeSchedulerConfig`,`kubeControllerManagerConfig`, o`kubeApiServerConfig`.

| Componente | Parametro | Valori supportati | Predefinita | Richiede un piano di controllo fornito |
|---|---|---|---|---|
| kube-scheduler | `nodeResourcesFit.scoringStrategy` | `LeastAllocated`, `MostAllocated` | `LeastAllocated`, con e `cpu: 1` `memory: 1` | No |
| kube-controller-manager | `horizontalPodAutoscalerControllerConfig.horizontalPodAutoscalerSyncPeriod` | Da `10s` a `15s` | `15s`(secondi) | Sì |
| kube-controller-manager | `podGcControllerConfig.terminatedPodGcThreshold` | Da `10000` a `12500` | `12500` | Sì |
| server kube-api | `eventTtl` | Da `10m` a `60m` | `60m`(minuti) | No |
| server kube-api | `serviceNodePortRange` | `minPort` tra `maxPort` e `10260` `32767` | `minPort: 30000`, `maxPort: 32767` | No |

I valori predefiniti e supportati in questo argomento si applicano alla versione di Kubernetes (EKS v1.31 e successive) disponibile al momento della pubblicazione e potrebbero cambiare nelle versioni successive. L'`DescribeClusterVersions`operazione riporta i valori predefiniti e supportati correnti per ogni parametro e versione di Kubernetes, quindi usala come fonte di verità se gestisci cluster su più versioni o automatizzi la configurazione dei cluster. Per ulteriori informazioni, consulta the section called "Configura i parametri del piano di controllo".

Le sezioni seguenti descrivono ogni parametro, quando modificarlo e cosa considerare prima di procedere.

## Scheduler: le risorse del nodo si adattano

Lo scheduler assegna i pod ai nodi in due fasi. Prima filtra i nodi che possono eseguire un pod, quindi assegna un punteggio ai candidati rimanenti e posiziona il pod nel nodo con il punteggio più alto. Il `nodeResourcesFit` plugin verifica se un nodo dispone delle risorse richieste da un pod e assegna un punteggio ai nodi in base a una strategia di punteggio.

| Campo | Description | Valori supportati | Predefinita |
|---|---|---|---|
| `nodeResourcesFit.scoringStrategy.type` | La strategia utilizzata per assegnare un punteggio ai nodi in base all'allocazione delle risorse. | `LeastAllocated` , `MostAllocated` | `LeastAllocated` |
| `nodeResourcesFit.scoringStrategy.resources` | Le risorse considerate per l'assegnazione del punteggio, ciascuna con un peso relativo. | `cpu`,`memory`,, `nvidia.com/gpu`aws.amazon.com/neuron ,aws.amazon.com/neuroncore . Pesi da 1 a. `100` | `cpu: 1`, `memory: 1` |

`LeastAllocated`privilegia i nodi con una minore allocazione delle risorse, distribuendo i pod tra i nodi del cluster e lasciando spazio a ciascun nodo. Questo è il comportamento predefinito di Kubernetes ed è una buona scelta quando si desidera la capacità disponibile su ogni nodo per assorbire la crescita dei pod esistenti.

`MostAllocated`privilegia i nodi che dispongono già di una maggiore allocazione delle risorse, il che raggruppa i pod su un numero inferiore di nodi. Poiché i carichi di lavoro occupano una capacità totale inferiore, puoi eseguirli su un numero inferiore di nodi e ridurre la spesa di elaborazione. Nel tempo, questo comportamento di impacchettamento mantiene i nodi poco utilizzati liberi da nuovi carichi di lavoro, in modo che i pool di nodi che supportano il consolidamento possano rimuoverli.

EKS supporta le strategie `LeastAllocated` e. `MostAllocated` La `RequestedToCapacityRatio` strategia Kubernetes upstream non è supportata.

Ponderazione delle risorse

È possibile specificare facoltativamente un `resources` array con pesi personalizzati per determinare quali risorse sono più importanti nelle decisioni relative al punteggio. Ciò è utile quando una risorsa specifica è il vincolo del cluster. Ad esempio, nei cluster in cui gli acceleratori (GPU) sono la risorsa più scarsa, un peso `nvidia.com/gpu` superiore a CPU e memoria concentra i pod che richiedono acceleratori su nodi già parzialmente occupati.

I pesi sono relativi, non assoluti. L'impostazione `cpu: 100` e `memory: 1` non fa sì che lo scheduler ignori la memoria. Pesa la CPU 100 volte di più della memoria nella formula di punteggio. Se tutti i nodi candidati hanno una disponibilità di CPU identica, la CPU non li distingue più e il punteggio viene effettivamente assegnato alla memoria.

Omettere una risorsa è diverso dall'attribuirle un peso ridotto. Quando si specifica un `resources` array, vengono valutate solo le risorse elencate. Una risorsa tralasciata viene completamente esclusa dal calcolo. Ad esempio, `cpu: 100` se non vengono `memory` assegnati punteggi ai nodi, la sola CPU e la disponibilità di memoria non influiscono sul risultato. Per mantenere una risorsa nel calcolo riducendone l'influenza, elencala con un peso ridotto anziché ometterla.

La ponderazione di una risorsa acceleratrice, ad esempio, influisce `nvidia.com/gpu` solo sul punteggio dei pod che dichiarano effettivamente tale risorsa. `resources.requests` I pod che non richiedono acceleratori non sono influenzati dal peso dell'acceleratore.

Le tre risorse dell'acceleratore sono risorse estese di Kubernetes, ossia risorse a livello di nodo pubblicizzate a Kubernetes da un plugin anziché integrate. Vengono valutate solo quando un plug-in del dispositivo le pubblicizza su kubelet tramite l'API del plug-in del dispositivo. Le risorse rese disponibili su un nodo da un solo driver di dispositivo non sono visibili al `nodeResourcesFit` plug-in e non vengono valutate. Le risorse gestite tramite Dynamic Resource Allocation (DRA) sono pianificate da un plug-in separato e non fanno parte del `nodeResourcesFit` punteggio, pertanto l'attivazione del DRA non modifica il comportamento di questo parametro. Per ulteriori informazioni

sulla configurazione dei plug-in dei dispositivi NVIDIA, vedere NVIDIA DRA e plug-in del dispositivo. https://docs.aws.amazon.com/eks/latest/userguide/device-management-nvidia-dra-device-plugin.html Per ulteriori informazioni sulla configurazione dei dispositivi Neuron, vedi Gestione dei dispositivi Neuron. https://docs.aws.amazon.com/eks/latest/userguide/device-management-neuron.html

La strategia di punteggio è uno dei numerosi input utilizzati dallo scheduler per calcolare un punteggio per ciascun nodo. Per ulteriori informazioni su come lo scheduler filtra e valuta i nodi, vedi Scheduling Framework nella documentazione di Kubernetes.

Considerazioni per la strategia di punteggio

- I pod funzionanti non vengono spostati. Lo scheduler Kubernetes non riposiziona mai un pod già in esecuzione. La modifica della strategia di punteggio influisce solo sulle decisioni di pianificazione future e il posizionamento dei pod esistenti è permanente. Per ribilanciare i pod già in esecuzione, sfrattali o riavviali.
- Il comportamento di filtraggio non cambia. La strategia di punteggio influisce solo sulla fase di punteggio, in cui lo scheduler classifica i nodi in base alle preferenze. La fase di filtraggio, che determina se un pod può essere eseguito su un nodo, rimane invariata. Un pod che non si adatta a un nodo non è ancora programmato in nessuna delle due strategie.
- **`MostAllocated`**concentra il raggio di esplosione. L'impacchettamento dei carichi di lavoro su un numero inferiore di nodi significa che più pod vengono colpiti contemporaneamente se un nodo diventa malsano, un'istanza viene ritirata o una zona di disponibilità viene interrotta. In presenza di un elevato tasso di abbandono dei pod, anche i nodi densamente impacchettati si riempiono più velocemente, il che può lasciare i pod in stato mentre viene fornita nuova capacità. `Pending`
- Lo scheduler e la gestione dei nodi operano a diversi livelli. La strategia di punteggio influenza la posizione dei pod tra i nodi che possono già eseguirli. Non cambia il modo in cui EKS Auto Mode o Karpenter forniscono o rimuovono i nodi. Convalida il comportamento combinato per il tuo carico di lavoro prima di modificare la configurazione.

## Controller manager: periodo di sincronizzazione Horizontal Pod Autoscaler

Il controller manager esegue i controller Kubernetes che indirizzano lo stato del cluster verso lo stato desiderato, incluso il controller Horizontal Pod Autoscaler (HPA). In ogni ciclo, il controller HPA recupera le metriche per ogni `HorizontalPodAutoscaler` oggetto, calcola il numero di repliche desiderato e aggiorna il carico di lavoro di destinazione se il conteggio è cambiato.

| Campo | Description | Valori supportati | Predefinita |
|---|---|---|---|
| `horizontalPodAutoscalerControllerConfig.horizontalPodAutoscalerSyncPeriod` | Con quale frequenza il controller HPA valuta le decisioni di scalabilità. | Da `10s` a `15s` | `15s` |

Riducendo il periodo di sincronizzazione, i carichi di lavoro si scalano subito dopo l'aumento del carico, anziché attendere un ciclo completo prima di aggiungere capacità.

Per configurare questo parametro, il cluster deve trovarsi su Amazon EKS Provisioned Control Plane. La riduzione dell'intervallo aumenta la velocità con cui il controller HPA riconcilia tutti gli `HorizontalPodAutoscaler` oggetti del cluster, generando più richieste API. Ogni riconciliazione consuma almeno una richiesta API e una riconciliazione che modifica il numero di repliche ne consuma altre due. I cluster Control Plane con provisioning preallocano la capacità del piano di controllo che è sempre pronta per carichi di lavoro impegnativi, quindi sono dimensionati per assorbire quel carico aggiuntivo. Per ulteriori informazioni, consulta the section called "Crea un piano di controllo predisposto".

Influenza sul numero di oggetti HPA supportati dal cluster

- La riduzione del periodo di sincronizzazione riduce il numero di **`HorizontalPodAutoscaler`** oggetti che il piano di controllo può riconciliare nei tempi previsti, poiché il controller ha meno tempo per lavorare sulla stessa coda. Riducendo il periodo da `15s` a si `10s` riduce il numero di oggetti supportati di circa un terzo. Prima di abbreviare il periodo di sincronizzazione, conta gli `HorizontalPodAutoscaler` oggetti nel cluster e conferma che il periodo più breve supporti ancora quel conteggio nel tuo livello di scalabilità:

  ```
  kubectl get hpa --all-namespaces --no-headers | wc -l
  ```

- EKS non convalida il periodo di sincronizzazione rispetto al numero di oggetti HPA. La modifica della configurazione ha esito positivo anche se il cluster ha già più `HorizontalPodAutoscaler` oggetti di quelli supportati dal periodo più breve. Verifica tu stesso il conteggio prima di apportare la modifica.
- Il superamento del conteggio supportato degrada automaticamente la scalabilità automatica in modo silenzioso. Se il controller non è in grado di elaborare tutti gli oggetti entro il periodo,

alcuni oggetti non vengono riconciliati nei tempi previsti. EKS non emette un allarme o un evento Kubernetes per questa condizione e il sintomo è la scalabilità automatica che risponde più lentamente del previsto, l'opposto dell'effetto previsto. Se osservi un ridimensionamento ritardato dopo aver ridotto il periodo di sincronizzazione, riporta il parametro al valore predefinito di. `15s`

- Il periodo di sincronizzazione si applica a tutti gli oggetti HPA nel cluster.  Non puoi impostare periodi di sincronizzazione diversi per oggetti o namespace diversi.

## Controller manager: soglia di raccolta dei rifiuti nei pod terminata

Il controller manager esegue il pod garbage collector terminato (il controller pod GC). Questo controller elimina i pod terminati, ovvero i pod in `Failed` fase `Succeeded` or, dopo che il numero di pod terminati nel cluster supera una soglia. Il parametro imposta tale `terminatedPodGcThreshold` soglia.

| Campo | Description | Valori supportati | Predefinita |
|---|---|---|---|
| `podGcControllerConfig.terminatedPodGcThreshold` | Il numero di pod terminati che possono esistere prima che il Garbage Collector dei pod terminati inizi a eliminare i pod terminati. | Da `10000` a `12500` | `12500` |

Il garbage collector funziona con un ciclo fisso di 20 secondi. Quando si abbassa la soglia, il raccoglitore inizia a eliminare forzatamente i pod terminati più vecchi nel ciclo successivo fino a quando il conteggio non raggiunge la nuova soglia.

Per configurare questo parametro, il cluster deve trovarsi su Amazon EKS Provisioned Control Plane. L'abbassamento della soglia aumenta il lavoro di garbage collection eseguito dal controller sul database del cluster (ecc.). Ogni pass di raccolta diventa idoneo per interrogare, elaborare ed eliminare altri pod terminati. I cluster Provisioned Control Plane preallocano la capacità del piano di controllo che è sempre pronta per carichi di lavoro impegnativi, in modo da poter assorbire quel carico aggiuntivo. Per ulteriori informazioni, consulta the section called “Crea un piano di controllo predisposto”.

Considerazioni sulla soglia terminata per la raccolta dei rifiuti nei pod

- L'abbassamento della soglia elimina immediatamente i pod terminati in eccesso. Se si abbassa la soglia (ad esempio, da `12500` a`10000`), il Garbage Collection Controller avvia l'eliminazione forzata dei pod terminati più vecchi nel ciclo successivo. Il controller continua finché il numero di pod terminati non raggiunge la nuova soglia. La riduzione non è graduale.
- La soglia si applica a tutti i pod terminati indipendentemente dal proprietario. Influisce sui pod in `Failed` fase `Succeeded` o indipendentemente dal fatto che siano di proprietà di un Job CronJob, o Deployment o siano autonomi. In pratica, Job e CronJob pod sono i responsabili più comuni del numero di pod terminati.
- L'abbassamento della soglia riduce la finestra di debug. I pod completati e falliti scompaiono `kubectl get pods` e `kubectl logs` prima. L'automazione che ispeziona i codici di uscita o i log dei Job pod finiti ha una finestra operativa più piccola.
- La soglia è un'impostazione globale del cluster. Non è possibile configurarlo per namespace o per Job. Per controllare il ciclo di vita dei pod di un singolo Job, usali su quel Job. `ttlSecondsAfterFinished`

## Server API: conservazione degli eventi

Il server API è il front-end per il piano di controllo di Kubernetes. Serve l'API Kubernetes e mantiene lo stato del cluster nel database del cluster (etcd). Kubernetes registra gli eventi per descrivere ciò che accade nel cluster, ad esempio decisioni sulla pianificazione dei pod, prelievi di immagini, errori nei controlli di integrità e azioni di ridimensionamento.

| Campo | Description | Valori supportati | Predefinita |
|---|---|---|---|
| `eventTtl` | Per quanto tempo il server API conserva gli eventi Kubernetes prima di eliminarli. | Da `10m` a `60m` | `60m` |

I cluster che eseguono carichi di lavoro ad alto tasso di abbandono, come lavori batch su larga scala, carichi di lavoro AI, pipeline e frequenti, accumulano rapidamente migliaia di eventi. CI/CD CronJobs Ogni evento conservato consuma uno spazio del database del cluster che compete con gli oggetti necessari per l'esecuzione del cluster e un'ampia raccolta di eventi rende le operazioni relative agli elenchi dei server API più costose da gestire.

La riduzione della conservazione degli eventi consente di cancellare più rapidamente questi dati diagnostici di breve durata, riducendo la pressione sullo storage del database del cluster e migliorando i tempi di risposta del server API per le query che richiedono un numero elevato di eventi.

Un periodo di conservazione più breve è una buona soluzione quando:

- Il cluster esegue batch CI/CD, AI o CronJob carichi di lavoro che generano un volume elevato di eventi.
- Osservi lo storage del database del cluster crescere verso il limite.
- Ti affidi a un sistema esterno per acquisire gli eventi in modo duraturo e non dipendi da esso `kubectl get events` per il debug cronologico.

Considerazioni sulla conservazione degli eventi

- Una modifica si applica solo ai nuovi eventi. Kubernetes imposta la scadenza di un evento quando l'evento viene creato. Gli eventi già esistenti mantengono il periodo di conservazione in vigore al momento della loro creazione e scadono in base a tale pianificazione. La riduzione `eventTtl` non riduce la durata degli eventi già presenti nel database del cluster, pertanto la riduzione dello spazio di archiviazione ha effetto gradualmente man mano che gli eventi esistenti scadono.
- Gli eventi eliminati non possono essere recuperati. Dopo che Kubernetes ha rimosso un evento, questo viene eliminato definitivamente. Se riduci la conservazione oltre il previsto e perdi la cronologia degli eventi, non c'è modo di ripristinarla. Verifica che tutto ciò da cui dipendi per la risoluzione dei problemi venga acquisito all'esterno del cluster prima di abbreviare questo valore.
- Gli eventi possono persistere leggermente oltre il periodo configurato. In alcune condizioni, la scadenza di un evento può essere estesa oltre il valore configurato a causa del rinnovo del leasing etcd che potrebbe avvenire durante l'elezione del leader del piano di controllo.
- Finestra di debug ridotta. Un periodo di conservazione più breve restringe la finestra mostrata da `kubectl get events` e`kubectl describe`. Gli strumenti di monitoraggio che raccolgono gli eventi dal cluster hanno meno dati disponibili. Scegli un valore che bilanci l'efficienza dello storage con il flusso di lavoro di debug.
- L'impostazione è a livello di cluster. La conservazione si applica a tutti gli eventi, inclusa la pianificazione dei pod, le condizioni dei nodi e gli eventi di scalabilità, in ogni namespace. Non è possibile impostare periodi di conservazione diversi per namespace.

## Server API: intervallo di porte del nodo di servizio

Kubernetes assegna una porta da questo intervallo su ogni nodo per ogni servizio che ne ha bisogno. Ciò include i servizi di tipo `NodePort` e, per impostazione predefinita, i servizi di tipo. `LoadBalancer`

| Campo | Description | Valori supportati | Predefinita |
|---|---|---|---|
| `serviceNodePortRange.minPort` | La porta più bassa dell'intervallo. | Da `10260` a `32767` | `30000` |
| `serviceNodePortRange.maxPort` | La porta più alta dell'intervallo. | Da `10260` a `32767` | `32767` |

`minPort`deve essere minore o uguale a`maxPort`. Amazon EKS rifiuta una configurazione in cui `minPort` è maggiore di`maxPort`.

Modificando l'intervallo, puoi allineare l'allocazione delle porte dei nodi con le politiche di rete e firewall già applicate dalla tua organizzazione. L'ampliamento dell'intervallo aumenta anche il numero di servizi che un singolo cluster è in grado di supportare. Questo parametro è particolarmente utile durante le migrazioni. Le applicazioni che passano ad Amazon EKS e i client che le chiamano spesso si aspettano servizi su porte fisse specifiche. Quando tali porte non rientrano nell'intervallo predefinito, le opzioni usuali sono modificare l'applicazione o posizionarla davanti a un proxy. L'allineamento dell'intervallo con le porte già utilizzate dalle applicazioni elimina il lavoro, quindi è possibile spostare i carichi di lavoro su EKS senza riscriverli o aggiungere componenti di rete da mantenere.

Perché l'intervallo è limitato a 10260 e 32767

Il limite inferiore di `10260` mantiene l'`NodePort`allocazione libera dalle porte già utilizzate dai componenti del sistema Kubernetes sui nodi, tra cui kubelet health port (`10248`) e kube-proxy health check port (). `10256`

Il limite superiore di `32767` mantiene l'intervallo libero dall'intervallo di porte temporanee di Linux, che in genere inizia alle. `32768` Se a `NodePort` rientrasse nell'intervallo temporaneo, il kernel potrebbe selezionare quella porta per una connessione in uscita dal nodo e entrare in conflitto con il servizio.

Considerazioni sull'intervallo di porte del nodo di servizio

- I servizi esistenti mantengono le porte assegnate. Se riduci l'intervallo, i servizi che hanno già una porta al di fuori del nuovo intervallo continuano a funzionare e kube-proxy continua a indirizzare il traffico verso di essi. Amazon EKS non riassegna le porte per i servizi esistenti quando l'intervallo cambia.
- La ricreazione di un servizio ne rialloca la porta. Se un servizio che contiene una porta fuori portata viene eliminato e ricreato, tale porta non può più essere assegnata. Pianificate questa operazione prima di restringere l'intervallo da cui dipendono i servizi esistenti, in particolare se il processo di distribuzione ricrea i servizi anziché aggiornarli.
- Le nuove allocazioni al di fuori dell'intervallo vengono rifiutate. La creazione o l'aggiornamento di un servizio che richiede una porta al di fuori dell'intervallo configurato non riesce con un errore di convalida dal server API.
- Anche le porte specificate in modo esplicito vengono convalidate. Se un servizio specifica direttamente un `nodePort` valore anziché lasciare che Kubernetes ne assegni uno, quella porta deve rientrare nell'intervallo configurato. Una richiesta di porta statica al di fuori dell'intervallo viene rifiutata, anche se la stessa porta era valida in un intervallo più ampio configurato in precedenza.
- L'intervallo è esteso a tutto il cluster. Non è possibile configurare intervalli diversi per namespace diversi.

Prima di modificare questo parametro, verificate che i gruppi di sicurezza e gli ACL di rete consentano il traffico sul nuovo intervallo e che l'intervallo non sia in conflitto con le porte utilizzate da altri software sui nodi.

## Considerazioni

Prima di configurare i parametri avanzati del piano di controllo, consulta quanto segue.

- Cluster-wide ambito: i parametri del piano di controllo si applicano all'intero cluster e a tutti i carichi di lavoro in esecuzione su di esso. Non è possibile applicarli a singoli namespace o carichi di lavoro. Verifica le modifiche ai parametri in un cluster non di produzione prima di applicarle alla produzione.
- Il Provisioned Control Plane è necessario per il periodo di sincronizzazione di Horizontal Pod Autoscaler e la soglia di garbage collection del pod è terminata. I **`terminatedPodGcThreshold`** parametri **`horizontalPodAutoscalerSyncPeriod`** and sono disponibili solo sui cluster che utilizzano Amazon EKS Provisioned Control Plane. Amazon EKS limita i parametri che aumentano

sostanzialmente il consumo di risorse del piano di controllo ai cluster con capacità del piano di controllo preallocata. L'impostazione di uno dei due parametri su un cluster in modalità piano di controllo standard non riesce. Per utilizzarli, sposta innanzitutto il cluster su un livello di scalabilità del Provisioned Control Plane. Per ulteriori informazioni, consulta the section called "Crea un piano di controllo predisposto".

- Restrizione di uscita per il periodo di sincronizzazione di Horizontal Pod Autoscaler e termine della soglia di garbage collection del pod: se `horizontalPodAutoscalerSyncPeriod` o `terminatedPodGcThreshold` è impostato su un valore diverso da quello predefinito, non puoi spostare il piano di controllo del cluster dalla modalità Provisioned alla modalità Standard. Per tornare alla modalità Standard, prima imposta entrambi i parametri sui valori predefiniti (`15se12500`), quindi modifica il livello di scala del piano di controllo su. `standard`
- Ritorno ai valori predefiniti: Amazon EKS non fornisce un'operazione di ripristino dedicata e l'omissione di un campo da un aggiornamento ne mantiene il valore corrente anziché cancellarlo. Per riportare un parametro ai valori predefiniti, impostalo esplicitamente sul valore predefinito. Recupera il valore predefinito con la versione `DescribeClusterVersions` di Kubernetes eseguita dal cluster. Per ulteriori informazioni, consulta the section called "Configura i parametri del piano di controllo".
- Semantica degli aggiornamenti: gli aggiornamenti si fondono con la configurazione esistente. Solo i campi specificati vengono modificati e i campi omessi mantengono i valori correnti. Ciò vale sia per tutti i componenti che all'interno di un singolo componente. Ad esempio, un aggiornamento che specifica solo la configurazione dello scheduler lascia invariata la configurazione del controller manager e del server API.
- Visualizzazione della configurazione corrente: l'`describe-cluster`operazione restituisce la configurazione completa in esecuzione sul piano di controllo, inclusi i parametri non personalizzati e i relativi valori predefiniti.
- I valori predefiniti e supportati possono cambiare tra le versioni di Kubernetes: i valori documentati in questo argomento si applicano alle versioni di Kubernetes disponibili al momento della pubblicazione. `DescribeClusterVersions`Da utilizzare per recuperare i valori predefiniti e supportati correnti per ogni parametro e versione di Kubernetes. Consulta the section called "Configura i parametri del piano di controllo".
- I cluster esistenti sono invariati: Amazon EKS non modifica il comportamento dei cluster esistenti. Tutti i cluster continuano a funzionare con i valori dei parametri predefiniti finché non si imposta esplicitamente un parametro.
- Le modifiche non vengono applicate istantaneamente: una modifica della configurazione non è effettiva al momento `UpdateClusterConfig` della restituzione. Amazon EKS applica la

nuova configurazione tramite un aggiornamento continuo del piano di controllo, quindi aspettati alcuni minuti prima che la modifica abbia effetto completo. Il cluster ritorna allo `ACTIVE` stato una volta completato l'aggiornamento. È possibile tenere traccia dei progressi utilizzando l'DescribeUpdateoperazione o bloccarla fino al completamento della modifica utilizzando. `aws eks wait cluster-active`

- Verificabilità: Amazon EKS convalida ogni configurazione prima di applicarla e registra le modifiche alla configurazione. AWS CloudTrail
- Supporto per gli strumenti: la configurazione avanzata del piano di controllo Kubernetes è disponibile tramite eksctl, AWS CLI Console di gestione AWS, API Amazon EKS e CDK al momento del lancio. AWS CloudFormation AWS Il supporto per AWS Controllers for Kubernetes (ACK) e Terraform sarà presto disponibile.
- Supporto della versione Kubernetes: la configurazione avanzata del piano di controllo Kubernetes è supportata su cluster nuovi ed esistenti che eseguono Kubernetes versione 1.31 o successiva.
- AWS Supporto per regione: la configurazione avanzata del piano di controllo Kubernetes è disponibile in tutte le regioni AWS commerciali, nelle regioni (Stati Uniti) e nelle regioni della Cina in cui è disponibile Amazon EKS. AWS GovCloud AWS
- Prezzi: non sono previsti costi aggiuntivi per la configurazione dei parametri del piano di controllo. L'utilizzo `horizontalPodAutoscalerSyncPeriod` richiede il Provisioned Control Plane, che viene fatturato alla tariffa oraria per il livello di scalabilità. Per ulteriori informazioni, consultare Prezzi di Amazon EKS.

## Fasi successive

- the section called "Configura i parametri del piano di controllo"— Imposta e visualizza i parametri del piano di controllo utilizzando la AWS CLI e. Console di gestione AWS
- the section called "Crea un piano di controllo predisposto"— capacità Pre-allocate del piano di controllo per prestazioni prevedibili e elevate.

## Configurazione dei parametri avanzati del piano di controllo Kubernetes

Questa guida illustra l'impostazione e la visualizzazione dei parametri avanzati del piano di controllo Kubernetes utilizzando la CLI e. AWS Console di gestione AWS Per una spiegazione di ciascun parametro e del relativo effetto sul cluster, consulta. the section called "Configurazione avanzata del piano di controllo"

## Prerequisiti

Prima di iniziare, assicurati di avere:

- Un cluster Amazon EKS con Kubernetes versione 1.31 o successiva: la configurazione avanzata del piano di controllo Kubernetes è supportata su cluster nuovi ed esistenti che eseguono Kubernetes versione 1.31 o successiva. Tutti i parametri di questo argomento condividono questa versione minima.
- AWS CLI: uno strumento a riga di comando per lavorare con AWS servizi, incluso Amazon EKS. Per ulteriori informazioni, consulta la Guida utente all'installazione nell'interfaccia a riga di AWS comando. Dopo aver installato la AWS CLI, ti consigliamo di configurarla anche tu. Per ulteriori informazioni, consulta Configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.
- Autorizzazioni IAM richieste: il principale IAM che stai utilizzando deve disporre delle autorizzazioni per descrivere e aggiornare i cluster Amazon EKS. Per verificare l'entità corrente, esegui il comando seguente:

```
aws sts get-caller-identity
```

> **Note**
>
> Ti consigliamo di completare la procedura descritta in questo argomento in una shell Bash. In alternativa, puoi apportare alcune modifiche alla tua shell per alcuni comandi di script, come i caratteri di continuazione della riga, e per il modo in cui le variabili vengono impostate e utilizzate. Inoltre, le regole di escape e di utilizzo delle virgolette per la shell (interprete di comandi) potrebbero essere diverse. Per ulteriori informazioni, vedere Utilizzo delle virgolette con stringhe nella AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

Imposta il nome e la regione del cluster preferiti prima di eseguire i comandi in questo argomento:

```
export CLUSTER=my-eks-cluster
export AWS_REGION=us-west-2
```

## Valori supportati

Amazon EKS convalida ogni parametro rispetto ai seguenti intervalli e rifiuta una richiesta al di fuori di essi.

| Parametro del componente | Valori supportati | Predefinita |
|---|---|---|
| `kubeSchedulerConfig.nodeResourcesFit.scoringStrategy.type` | `LeastAllocated` , `MostAllocated` | `LeastAllocated` |
| `kubeSchedulerConfig.nodeResourcesFit.scoringStrategy.resources[].weight` | Da `1` a `100` | `cpu: 1`, `memory: 1` |
| `kubeControllerManagerConfig.horizontalPodAutoscalerControllerConfig.horizontalPodAutoscalerSyncPeriod` | Da `10s` a `15s` | `15s` |
| `kubeControllerManagerConfig.podGcControllerConfig.terminatedPodGcThreshold` | Da `10000` a `12500` | `12500` |
| `kubeApiServerConfig.eventTtl` | Da `10m` a `60m` | `60m` |
| `kubeApiServerConfig.serviceNodePortRange.minPort` | Da `10260` a `32767` | `30000` |

| Parametro del componente | Valori supportati | Predefinita |
| --- | --- | --- |
| `kubeApiServerConfig.serviceNodePortRange.maxPort` | Da `10260` a `32767` | `32767` |

Questi valori si applicano alle versioni di Kubernetes disponibili al momento della pubblicazione e possono cambiare nelle versioni successive. Per recuperare i valori correnti per una versione specifica, consulta la sezione seguente.

## Trova i valori predefiniti e supportati per una versione di Kubernetes

L'`DescribeClusterVersions`operazione riporta il valore predefinito e i valori supportati per ogni parametro del piano di controllo, per ogni versione di Kubernetes. Leggi i valori di questa operazione anziché codificarli, in particolare se automatizzi la configurazione dei cluster o gestisci i cluster in diverse versioni di Kubernetes.

```
aws eks describe-cluster-versions --cluster-versions 1.35
```

La risposta include un `controlPlaneComponentConfig` oggetto, organizzato per componente del piano di controllo. Ogni parametro riporta un `defaultValue` `constraints` oggetto:

```
{
    "clusterVersions": [
        {
            "clusterVersion": "1.35",
            "clusterType": "eks",
            "defaultVersion": false,
            "releaseDate": "2026-01-23T00:00:00+00:00",
            "endOfStandardSupportDate": "2027-03-23T00:00:00+00:00",
            "endOfExtendedSupportDate": "2028-03-23T00:00:00+00:00",
            "status": "STANDARD_SUPPORT",
            "controlPlaneScalingTiers": [
                {
                    "tierName": "tier-xl",
                    "apiRequestConcurrency": 2000,
                    "podSchedulingRatePerSecond": 167,
                    "clusterDatabaseSizeGb": 16
                },
                {
```

```
                "tierName": "tier-2xl",
                "apiRequestConcurrency": 4000,
                "podSchedulingRatePerSecond": 283,
                "clusterDatabaseSizeGb": 16
            },
            {
                "tierName": "tier-4xl",
                "apiRequestConcurrency": 8000,
                "podSchedulingRatePerSecond": 400,
                "clusterDatabaseSizeGb": 16
            },
            {
                "tierName": "tier-8xl",
                "apiRequestConcurrency": 16000,
                "podSchedulingRatePerSecond": 400,
                "clusterDatabaseSizeGb": 16
            }
        ],
        "controlPlaneComponentConfig": {
            "kubeApiServerConfig": {
                "eventTtl": {
                    "defaultValue": "60m",
                    "constraints": {
                        "min": "10m",
                        "max": "60m"
                    }
                },
                "serviceNodePortRange": {
                    "defaultValue": {
                        "minPort": 30000,
                        "maxPort": 32767
                    },
                    "constraints": {
                        "minPort": {"min": 10260, "max": 32767},
                        "maxPort": {"min": 10260, "max": 32767}
                    }
                }
            },
            "kubeSchedulerConfig": {
                "nodeResourcesFit": {
                    "scoringStrategy": {
                        "defaultValue": {
                            "type": "LeastAllocated",
                            "resources": [
```

```
                                {"name": "cpu", "weight": 1},
                                {"name": "memory", "weight": 1}
                            ]
                        },
                        "constraints": {
                            "scoringStrategy": {
                                "allowedValues": ["LeastAllocated",
"MostAllocated"]
                            },
                            "resources": {
                                "name": {
                                    "allowedValues": [
                                        "cpu",
                                        "memory",
                                        "nvidia.com/gpu",
                                        "aws.amazon.com/neuron",
                                        "aws.amazon.com/neuroncore"
                                    ]
                                },
                                "weight": {"min": 1, "max": 100}
                            }
                        }
                    }
                },
                "kubeControllerManagerConfig": {
                    "horizontalPodAutoscalerControllerConfig": {
                        "horizontalPodAutoscalerSyncPeriod": {
                            "defaultValue": "15s",
                            "constraints": {
                                "min": "10s",
                                "max": "15s"
                            }
                        }
                    },
                    "podGcControllerConfig": {
                        "terminatedPodGcThreshold": {
                            "defaultValue": 12500,
                            "constraints": {
                                "min": 10000,
                                "max": 12500
                            }
                        }
                    }
```

```
                }
            }
        }
    ]
}
```

La risposta precedente mostra le date di supporto della versione, i livelli di scalabilità del Provisioned Control Plane e i campi di configurazione del piano di controllo. Per ulteriori informazioni, consulta Amazon EKS DescribeClusterVersions API Reference.

Per restituire solo i campi di configurazione, utilizza l'`--query`opzione:

```
aws eks describe-cluster-versions --cluster-versions 1.35 \
--query 'clusterVersions[0].controlPlaneComponentConfig'
```

## Configurazione del piano di controllo — AWS CLI

Ogni componente del piano di controllo ha il proprio parametro:`--kube-scheduler-config`,`--kube-controller-manager-config`, e`--kube-api-server-config`. Specificate solo i componenti da personalizzare. Ogni parametro accetta JSON in linea.

Crea un cluster con i parametri del piano di controllo

Il comando seguente crea un cluster che conserva i nuovi eventi Kubernetes per 10 minuti invece dei 60 minuti predefiniti. I parametri che non specifichi utilizzano i loro valori predefiniti.

```
aws eks create-cluster \
--name "$CLUSTER" \
--kubernetes-version 1.35 \
--role-arn arn:aws:iam::111122223333:role/eks-service-role \
--resources-vpc-config subnetIds=subnet-abc123,subnet-def456,securityGroupIds=sg-xyz789
 \
--kube-api-server-config '{
  "eventTtl": "10m"
}'
```

Risposta:

```
{
    "cluster": {
```

```
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-west-2:111122223333:cluster/my-eks-cluster",
        "createdAt": 1709640000.0,
        "version": "1.35",
        "roleArn": "arn:aws:iam::111122223333:role/eks-service-role",
        "status": "CREATING",
        "kubeSchedulerConfig": {
            "nodeResourcesFit": {
                "scoringStrategy": {
                    "type": "LeastAllocated"
                }
            }
        },
        "kubeControllerManagerConfig": {
            "horizontalPodAutoscalerControllerConfig": {
                "horizontalPodAutoscalerSyncPeriod": "15s"
            }
        },
        "kubeApiServerConfig": {
            "eventTtl": "10m",
            "serviceNodePortRange": {
                "minPort": 30000,
                "maxPort": 32767
            }
        }
    }
}
```

Visualizza la configurazione del piano di controllo per un cluster

```
aws eks describe-cluster --name "$CLUSTER"
```

La risposta include `kubeSchedulerConfigkubeControllerManagerConfig`, e `kubeApiServerConfig` come campi di primo livello. Questi campi sono sempre presenti e mostrano la configurazione completa in esecuzione sul piano di controllo, inclusi i parametri non personalizzati e i relativi valori predefiniti.

Per restituire la configurazione per un singolo componente, usa l'`--query`opzione:

```
aws eks describe-cluster --name "$CLUSTER" \
--query 'cluster.kubeSchedulerConfig'
```

Risposta:

```
{
    "nodeResourcesFit": {
        "scoringStrategy": {
            "type": "LeastAllocated"
        }
    }
}
```

Configura la strategia di punteggio dello scheduler

Per impostazione predefinita, lo scheduler utilizza la `LeastAllocated` strategia, che distribuisce i pod tra i nodi. Il comando seguente lo modifica in`MostAllocated`, che preferisce i nodi che hanno già un'allocazione di risorse più elevata e impacchetta i pod su un numero inferiore di nodi.

```
aws eks update-cluster-config \
--name "$CLUSTER" \
--kube-scheduler-config '{
  "nodeResourcesFit": {
    "scoringStrategy": {
      "type": "MostAllocated"
    }
  }
}'
```

Risposta:

```
{
    "update": {
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "status": "InProgress",
        "type": "ControlPlaneComponentConfigUpdate",
        "params": [
            {
                "type": "kubeSchedulerConfig",
                "value": "{\"nodeResourcesFit\":{\"scoringStrategy\":{\"type\":
\"MostAllocated\"}}}"
            }
        ],
        "createdAt": 1709643600.0,
        "errors": []
```

```
    }
}
```

È inoltre possibile specificare un `resources` array per valutare le singole risorse nel calcolo del punteggio, utile quando il vincolo del cluster è una risorsa specifica. Per ulteriori informazioni, consulta the section called "Configurazione avanzata del piano di controllo".

Gli altri parametri del piano di controllo seguono lo stesso schema, ognuno con il proprio flag di componente: `--kube-api-server-config` per `eventTtl` e e `serviceNodePortRange` per il periodo di sincronizzazione di Horizontal Pod Autoscaler. `--kube-controller-manager-config` Per sapere cosa fa ogni parametro e cosa considerare prima di modificarlo, vedi. the section called "Configurazione avanzata del piano di controllo"

Aggiorna più di un componente alla volta

È possibile specificare più parametri dei componenti in una singola chiamata:

```
aws eks update-cluster-config \
--name "$CLUSTER" \
--kube-scheduler-config '{"nodeResourcesFit":{"scoringStrategy":
{"type":"MostAllocated"}}}' \
--kube-api-server-config '{"eventTtl":"30m"}'
```

Gli aggiornamenti si fondono con la configurazione esistente. Solo i campi specificati vengono modificati e i campi omessi mantengono i valori correnti. Il comando precedente lascia invariata la configurazione del controller manager.

Monitora l'aggiornamento della configurazione del piano di controllo

Una modifica della configurazione non è effettiva al momento della `update-cluster-config` restituzione. Amazon EKS applica la nuova configurazione tramite un aggiornamento continuo del piano di controllo, che richiede diversi minuti. Per bloccare fino al completamento della modifica:

```
aws eks wait cluster-active --name "$CLUSTER"
```

Questo comando viene restituito quando il cluster raggiunge `ACTIVE` lo stato, in genere entro alcuni minuti.

Per elencare gli aggiornamenti per il cluster:

```
aws eks list-updates --name "$CLUSTER"
```

Risposta:

```
{
    "updateIds": [
        "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    ]
}
```

Per visualizzare lo stato di un aggiornamento specifico:

```
aws eks describe-update --name "$CLUSTER" \
--update-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

Lo stato cambia da `InProgress` a `Successful` quando viene completato l'aggiornamento o a `Failed` se si verifica un errore. Il `errors` campo indica il motivo dell'errore.

## Configurazione del piano di controllo: Console di gestione AWS

Per configurare i parametri avanzati del piano di controllo Kubernetes durante la creazione di un cluster:

1. Aprire la Console Amazon EKS.
2. Scegli Crea cluster.
3. In Opzioni di configurazione, seleziona Configurazione personalizzata.
4. Scorri verso il basso fino a Configurazione del piano di  controllo.
5. Scegli  Abilita configurazione del piano di controllo.
6. Imposta i parametri che desideri configurare. I parametri che lasci invariati utilizzano i loro valori predefiniti.
7. Seleziona altre opzioni di configurazione del cluster in base alle esigenze. Nel passaggio finale, scegli  Crea cluster. Il completamento della creazione del cluster potrebbe richiedere alcuni minuti.

Per configurare i parametri avanzati del piano di controllo Kubernetes su un cluster esistente:

1. Aprire la Console Amazon EKS.
2. Scegli il cluster che desideri aggiornare.

3. Scegli la scheda Panoramica, quindi scorri verso il basso fino a Configurazione del piano di controllo.
4. Scegliete Gestisci, quindi selezionate Abilita configurazione del piano di controllo per impostare i parametri da modificare. Selezionare quindi Save changes (Salva modifiche).

# Prepararsi agli aggiornamenti delle versioni di Kubernetes e risolvere i problemi di configurazione errata con gli approfondimenti sui cluster

Gli approfondimenti sui cluster di Amazon EKS rilevano i problemi e forniscono consigli per risolverli e gestire il cluster. Ogni cluster Amazon EKS è soggetto a controlli automatici e ricorrenti basati su un elenco di approfondimenti curato da Amazon EKS. Questi controlli sugli approfondimenti sono completamente gestiti da Amazon EKS e offrono consigli su come affrontare eventuali esiti.

## Tipi di approfondimenti del cluster

- Approfondimenti sulla configurazione: identificano le configurazioni errate nella configurazione di nodi ibridi EKS che potrebbero compromettere la funzionalità del cluster o dei carichi di lavoro.
- Approfondimenti sugli aggiornamenti: identificano i problemi che potrebbero influire sulla capacità di aggiornamento a nuove versioni di Kubernetes.
- Informazioni sulla preparazione al rollback: identifica i problemi che potrebbero influire sulla capacità di ripristinare una versione precedente di Kubernetes dopo un aggiornamento.

## Considerazioni

- Frequenza: Amazon EKS aggiorna le informazioni sul cluster ogni 24 ore oppure è possibile aggiornarle manualmente per visualizzare lo stato più recente. Ad esempio, puoi aggiornare manualmente le informazioni sul cluster dopo aver risolto un problema per vedere se il problema è stato risolto.
- Autorizzazioni: Amazon EKS crea automaticamente una voce di accesso al cluster per le informazioni sui cluster in ogni cluster EKS. Questa voce concede l'autorizzazione a EKS per visualizzare le informazioni relative al cluster. Amazon EKS utilizza queste informazioni per generare approfondimenti. Per ulteriori informazioni, consulta the section called "AmazonEKSClusterInsightsPolicy".

- Disponibilità del rollback readiness: le informazioni sulla preparazione al rollback sono disponibili solo per i cluster che sono stati aggiornati negli ultimi 7 giorni. Dopo la scadenza del periodo di idoneità al rollback di 7 giorni, queste informazioni non vengono più generate per il cluster.

## Casi d'uso

Gli approfondimenti sui cluster in Amazon EKS forniscono controlli automatizzati per aiutare a mantenere l'integrità, l'affidabilità e la configurazione ottimale dei cluster Kubernetes. Di seguito sono riportati i principali casi d'uso degli approfondimenti sui cluster, tra cui la preparazione all'aggiornamento e la risoluzione dei problemi di configurazione.

### Approfondimenti sugli aggiornamenti

Gli approfondimenti sugli aggiornamenti sono un tipo specifico di controlli approfonditi all'interno degli approfondimenti del cluster. Questi controlli restituiscono informazioni relative alla preparazione all'aggiornamento della versione di Kubernetes. Amazon EKS esegue controlli approfonditi sugli aggiornamenti su ogni cluster EKS.

> **Important**
>
> Amazon EKS ha temporaneamente eseguito il rollback di una funzionalità che richiedeva l'utilizzo di un flag `--force` per aggiornare il cluster in caso di determinati problemi di approfondimenti sul cluster. Per ulteriori informazioni, consulta Ripristino temporaneo dell'applicazione delle informazioni sull'aggiornamento all'aggiornamento della versione del cluster su. GitHub
> Per ulteriori informazioni sull'aggiornamento del cluster, consulta the section called "Fase 3: aggiornamento del piano di controllo del cluster".

Prima di aggiornare la versione di Kubernetes del cluster, puoi utilizzare la scheda Approfondimenti sugli aggiornamenti della dashboard di osservabilità nella console Amazon EKS. Se il cluster ha identificato problemi, consultali e apporta le correzioni appropriate. I problemi includono collegamenti alla documentazione di Amazon EKS e Kubernetes. Dopo aver risolto il problema, aggiornare gli aggiornamenti sui cluster su richiesta per recuperare gli approfondimenti più recenti. Se tutti i problemi sono stati risolti, aggiorna il cluster.

Amazon EKS restituisce approfondimenti riguardanti la preparazione all'aggiornamento della versione di Kubernetes. Gli approfondimenti sugli aggiornamenti identificano possibili problemi che

possono influire sugli aggiornamenti del cluster di Kubernetes. Questo riduce l'impegno richiesto agli amministratori nella preparazione degli aggiornamenti e aumenta l'affidabilità delle applicazioni sulle versioni più recenti di Kubernetes. I cluster vengono scansionati automaticamente da Amazon EKS in base a un elenco di possibili aggiornamenti di versione di Kubernetes che influiscono sui problemi. Amazon EKS aggiorna frequentemente l'elenco dei controlli sugli approfondimenti in base alle revisioni delle modifiche apportate in ogni versione di Kubernetes.

Gli approfondimenti sugli aggiornamenti offerti da Amazon EKS velocizzano il processo di test e verifica per le nuove versioni. Inoltre, consentono agli amministratori dei cluster e agli sviluppatori di applicazioni di avvalersi delle funzionalità di Kubernetes più recenti, evidenziando eventuali problemi e offrendo consigli per porvi rimedio.

## Approfondimenti sulla configurazione

Gli approfondimenti sui cluster di EKS analizzano automaticamente i cluster di Amazon EKS con nodi ibridi per identificare problemi di configurazione che compromettono la comunicazione tra piano di controllo e webhook di Kubernetes, comandi kubectl come exec e logs, e altro ancora. Gli approfondimenti sulla configurazione evidenziano i problemi e forniscono consigli per la correzione, accelerando i tempi per una configurazione di nodi ibridi perfettamente funzionante.

## Informazioni sulla preparazione al rollback

I Rollback Readiness Insights sono un tipo specifico di controlli approfonditi all'interno di Cluster Insights. Questi controlli restituiscono informazioni relative alla preparazione al rollback delle versioni di Kubernetes. Amazon EKS esegue controlli approfonditi sulla preparazione al rollback sui cluster che sono stati aggiornati negli ultimi 7 giorni. Gli approfondimenti sulla preparazione al rollback sono controlli puntuali: riflettono lo stato del cluster al momento della valutazione, non in modo continuo.

Dopo aver aggiornato la versione di Kubernetes del cluster, puoi utilizzare la scheda Upgrade insights della dashboard di osservabilità nella console Amazon EKS per esaminare le informazioni sulla preparazione al rollback. Se il cluster ha identificato problemi, consultali e apporta le correzioni appropriate. I problemi includono collegamenti alla documentazione di Amazon EKS e Kubernetes. Dopo aver risolto il problema, aggiornare gli aggiornamenti sui cluster su richiesta per recuperare gli approfondimenti più recenti. Se tutti i problemi sono stati risolti, puoi procedere con il rollback.

Amazon EKS restituisce informazioni relative alla preparazione al rollback delle versioni di Kubernetes nell'ambito della categoria. `ROLLBACK_READINESS` Gli approfondimenti sulla preparazione al rollback identificano possibili problemi che potrebbero influire sul rollback di una versione del cluster Kubernetes. Ciò riduce al minimo lo sforzo che gli amministratori dedicano

alla preparazione dei rollback e aumenta l'affidabilità del ripristino a una versione precedente. I cluster vengono scansionati automaticamente da Amazon EKS in base a un elenco di possibili problemi che influiscono sul rollback, tra cui la compatibilità dell'utilizzo dell'API (che copre i controlli di incompatibilità durante la gradazione della versione dell'API in cui viene rimossa una versione API precedente, nuove risorse che non esistono nella versione desiderata e nuove modifiche ai campi o all'enum), lo stato del cluster, l'inclinazione delle versioni di kubelet e kube-proxy, la compatibilità dei componenti aggiuntivi gestiti da EKS e, per i cluster in modalità automatica, l'interruzione controlli di budget e annotazioni.

> **Note**
>
> Gli approfondimenti sulla preparazione al rollback controllano solo le versioni dei EKS-managed componenti aggiuntivi. Per i componenti aggiuntivi autogestiti, o se avete sostituito la versione di un componente aggiuntivo gestito al di fuori del ciclo di vita del componente aggiuntivo EKS, Insights non rileva le incompatibilità delle versioni. L'utente è responsabile della convalida della compatibilità di tali componenti aggiuntivi con la versione di destinazione prima del rollback.

Gli approfondimenti con stato ERROR o UNKNOWN bloccano il rollback fino alla risoluzione del problema. Puoi utilizzare il `--force` flag per aggirare i controlli approfonditi se scegli di procedere a tuo rischio e pericolo. Gli approfondimenti con lo stato WARNING sono indicativi e non impediscono il rollback. Per ulteriori informazioni, consulta the section called “Ripristina la versione di Kubernetes”.

## Nozioni di base

Per visualizzare l'elenco dei controlli approfonditi eseguiti e gli eventuali problemi rilevanti identificati da Amazon EKS, puoi utilizzare l'interfaccia a riga di Console di gestione AWS comando, gli AWS AWS SDK e il funzionamento dell'API Amazon EKS`ListInsights`. Per iniziare, consulta the section called “Visualizzazione degli approfondimenti sui cluster”.

## Visualizzazione degli approfondimenti sui cluster

Amazon EKS fornisce tre tipi di informazioni: informazioni sulla configurazione, informazioni sugli upgrade e informazioni sulla preparazione al rollback. Gli approfondimenti sulla configurazione identificano le configurazioni errate nella configurazione di nodi ibridi EKS che potrebbero compromettere la funzionalità del cluster o dei carichi di lavoro. Gli approfondimenti sugli aggiornamenti identificano i problemi che potrebbero influire sulla capacità di aggiornamento a nuove

versioni di Kubernetes. Gli approfondimenti sulla preparazione al rollback identificano i problemi che potrebbero influire sulla capacità di ripristinare una versione precedente di Kubernetes dopo un aggiornamento.

Per visualizzare l'elenco dei controlli approfonditi eseguiti e gli eventuali problemi rilevanti identificati da Amazon EKS, puoi utilizzare l'interfaccia a riga di Console di gestione AWS comando, gli AWS AWS SDK e il funzionamento dell'API Amazon EKS`ListInsights`.

## Visualizzare approfondimenti sulla configurazione (console)

1. Aprire la Console Amazon EKS.
2. Dall'elenco dei cluster, scegli il nome del cluster Amazon EKS per il quale desideri visualizzare gli approfondimenti.
3. Scegli Monitora cluster.
4. Scegli la scheda Integrità del cluster.
5. Nella tabella Approfondimenti sulla configurazione, vedrai le seguenti colonne:
   - Nome: il controllo eseguito da Amazon EKS sul cluster.
   - Stato di approfondimento: un approfondimento con stato `Error` indica una configurazione errata che potrebbe influire sulla funzionalità del cluster. Un approfondimento con uno stato `Warning` indica che la configurazione non corrisponde all'approccio documentato, ma che la funzionalità del cluster potrebbe funzionare se configurata intenzionalmente. Un approfondimento con lo stato `Passing` indica che Amazon EKS non ha riscontrato alcun problema associato a questo controllo sull'approfondimento nel cluster.
   - Versione: la versione applicabile.
   - Ora dell'ultimo aggiornamento: l'ora in cui lo stato dell'approfondimento è stato aggiornato l'ultima volta per questo cluster.
   - Descrizione: informazioni tratte dal controllo sull'approfondimento, che include l'avviso e le azioni consigliate per la correzione.

## Visualizzare gli approfondimenti sugli aggiornamenti (console)

1. Aprire la Console Amazon EKS.
2. Dall'elenco dei cluster, scegli il nome del cluster Amazon EKS per il quale desideri visualizzare gli approfondimenti.
3. Scegli Monitora cluster.

4. Scegli la scheda Approfondimenti sugli aggiornamenti.
5. Per visualizzare i dati più recenti, scegli il pulsante Aggiorna approfondimenti e attendi il completamento dell'operazione di aggiornamento.
6. Nella tabella Approfondimenti sugli aggiornamenti, vedrai le seguenti colonne:
   - Nome: il controllo eseguito da Amazon EKS sul cluster.
   - Stato dell'approfondimento: un approfondimento che presenta lo stato "Errore" in genere indica che la versione di Kubernetes interessata è N+1 rispetto alla versione corrente del cluster, mentre lo stato "Avviso" indica che l'approfondimento si applica a una versione di Kubernetes futura N+2 o superiore. Un approfondimento con lo stato "Ammesso" indica che Amazon EKS non ha riscontrato alcun problema associato a questo controllo sull'approfondimento nel cluster. Lo stato dell'approfondimento "Sconosciuto" significa che Amazon EKS non è in grado di determinare se il cluster è interessato da questo controllo sull'approfondimento.
   - Versione: la versione di Kubernetes che l'approfondimento ha verificato per individuare possibili problemi.
   - Ora dell'ultimo aggiornamento: l'ora in cui lo stato dell'approfondimento è stato aggiornato l'ultima volta per questo cluster.
   - Ora dell'ultima transizione: l'ora in cui lo stato di questo approfondimento è cambiato l'ultima volta.
   - Descrizione: informazioni tratte dal controllo sull'approfondimento, che include l'avviso e le azioni consigliate per la correzione.

> **Note**
>
> Le informazioni sulla preparazione al rollback vengono visualizzate nella stessa scheda dopo aver eseguito un upgrade e rimangono visibili durante la finestra di idoneità al rollback di 7 giorni.

## Visualizza gli approfondimenti sui cluster (AWS CLI)

1. Utilizza gli approfondimenti per un cluster specificato per visualizzare i dati più recenti. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituiscilo *region-code* con il codice della tua AWS regione.
   - Sostituisci *my-cluster* con il nome del cluster.

```
aws eks start-insights-refresh --region region-code --cluster-name my-cluster
```

2. Esegui il comando seguente per tenere traccia dello stato di un aggiornamento di approfondimenti. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-insights-refresh --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "message": "Insights refresh is in progress",
    "status": "IN_PROGRESS",
    "startedAt": "2025-07-30T13:36:09-07:00"
}
```

3. Elenca gli approfondimenti per un cluster specificato. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituiscilo *region-code* con il codice AWS della tua regione.
   - Sostituisci *my-cluster* con il nome del cluster.

```
aws eks list-insights --region region-code --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
"insights":
    [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
```

```
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "name": "Kubelet version skew",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557309.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for kubelet versions of worker nodes in the
cluster to see if upgrade would cause non compliance with supported Kubernetes
kubelet version skew policy.",
            "insightStatus":
            {
                "status": "UNKNOWN",
                "reason": "Unable to determine status of node kubelet versions.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
```

```
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "Cluster health issues",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for any cluster health issues that prevent
successful upgrade to the next Kubernetes version on EKS.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No cluster health issues detected.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "EKS add-on version compatibility",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of installed EKS add-ons to ensure they
are compatible with the next version of Kubernetes. ",
            "insightStatus": { "status": "PASSING", "reason": "All installed EKS
add-on versions are compatible with next Kubernetes version."},
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "name": "kube-proxy version skew",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of kube-proxy in cluster to see if
upgrade would cause non compliance with supported Kubernetes kube-proxy version
skew policy.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "kube-proxy versions match the cluster control plane
version.",
            },
        },
        {
```

```
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEddddd",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE44444",
            "name": "Incompatible API usage",
            "category": "ROLLBACK_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of APIs that are not compatible
with the previous Kubernetes version. Rolling back your cluster before removing
incompatible API usage could cause data loss or application impact.",
            "insightStatus":
            {
                "status": "ERROR",
                "reason": "Incompatible API usage detected.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE55555",
            "name": "Kubelet version rollback compatibility",
            "category": "ROLLBACK_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557309.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for kubelet versions of worker nodes in the
cluster to see if rollback would cause non compliance with supported Kubernetes
kubelet version skew policy.",
            "insightStatus":
            {
```

```
                "status": "ERROR",
                "reason": "At least one node kubelet version matches the cluster
 control plane version.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE66666",
            "name": "EKS add-on version rollback compatibility",
            "category": "ROLLBACK_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of installed EKS add-ons to ensure they
 are compatible with the previous version of Kubernetes.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "All installed EKS add-on versions are compatible with
 previous Kubernetes version.",
            },
        },
    ],
"nextToken": null,
}
```

4. Esegui il comando seguente per visualizzare le informazioni dettagliate relative a un approfondimento. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituiscilo *region-code* con il codice AWS della tua regione.
   - *a1b2c3d4-5678-90ab-cdef-EXAMPLE22222*Sostituiscilo con un ID di approfondimento recuperato dall'elenco degli approfondimenti del cluster.
   - Sostituisci *my-cluster* con il nome del cluster.

   ```
   aws eks describe-insight --region region-code --id a1b2c3d4-5678-90ab-cdef-
   EXAMPLE22222 --cluster-name my-cluster
   ```

   Di seguito viene riportato un output di esempio.

   ```
   {
     "insight":
       {
   ```

```
      "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "name": "Kubelet version skew",
      "category": "UPGRADE_READINESS",
      "kubernetesVersion": "1.27",
      "lastRefreshTime": 1734557309.000,
      "lastTransitionTime": 1734557309.000,
      "description": "Checks for kubelet versions of worker nodes in the cluster
 to see if upgrade would cause non compliance with supported Kubernetes kubelet
 version skew policy.",
      "insightStatus":
        {
          "status": "UNKNOWN",
          "reason": "Unable to determine status of node kubelet versions.",
        },
      "recommendation": "Upgrade your worker nodes to match the Kubernetes version
 of your cluster control plane.",
      "additionalInfo":
        {
          "Kubelet version skew policy": "https://kubernetes.io/releases/version-
skew-policy/#kubelet",
          "Updating a managed node group": "https://docs.aws.amazon.com/eks/latest/
userguide/update-managed-node-group.html",
        },
      "resources": [],
      "categorySpecificSummary":
        { "deprecationDetails": [], "addonCompatibilityDetails": [] },
    },
}
```


L'output precedente mostra entrambi gli approfondimentiUPGRADE_READINESS. ROLLBACK_READINESS Le informazioni sulla preparazione al rollback sono disponibili solo per i cluster che sono stati aggiornati negli ultimi 7 giorni.

Filtra gli approfondimenti per categoria

Puoi filtrare gli approfondimenti per categoria per visualizzare solo un tipo specifico:

```
aws eks list-insights \
  --cluster-name my-cluster \
  --region region-code \
  --filter '{"categories": ["ROLLBACK_READINESS"]}'
```

Puoi anche filtrare per stato per visualizzare solo i problemi di blocco:

```
aws eks list-insights \
  --cluster-name my-cluster \
  --region region-code \
  --filter '{"categories": ["ROLLBACK_READINESS"], "statuses": ["ERROR"]}'
```

Visualizza i dettagli di Rollback Readiness Insight

Per informazioni sulla preparazione al rollback, il `describe-insight` comando restituisce informazioni simili sulle risorse interessate e sulle fasi di riparazione. Ad esempio:

```
{
    "clusterId": "73a0e91f-f016-4555-bb7c-177496c47c9d",
    "insight": {
        "category": "ROLLBACK_READINESS",
        "name": "API usage rollback compatibility",
        "kubernetesVersion": "1.32",
        "insightStatus": {
            "status": "ERROR",
            "reason": "Detected incompatible API objects with version rollback."
        },
        "resources": [
            {
                "kubernetesResourceUri": "/apis/networking.k8s.io/v1/servicecidrs/
kubernetes",
                "status": {
                    "status": "ERROR",
                    "reason": "networking.k8s.io/v1 is not compatible with Kubernetes
 version 1.32"
                }
            }
        ]
    }
}
```

Se trovi una panoramica sulla preparazione al rollback con`"status": "ERROR"`, devi risolvere il problema prima di eseguire il rollback oppure utilizzare il flag per aggirare i controlli di analisi approfondita`--force`. Per ulteriori informazioni sul processo di rollback, consulta. the section called "Ripristina la versione di Kubernetes"

Se un'analisi mostra `UNKNOWN` lo stato, EKS non è stata in grado di valutarla. Il rollback è bloccato fino a quando l'analisi non può essere valutata correttamente, oppure si utilizza il `--force` flag per aggirare i controlli di analisi.

# Aggiornamento del cluster esistente alla nuova versione di Kubernetes

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Quando è disponibile una nuova versione di Kubernetes in Amazon EKS, è possibile aggiornare il cluster Amazon EKS alla versione più recente.

> **Important**
>
> Dopo l'aggiornamento, puoi tornare alla versione precedente entro 7 giorni in caso di problemi. Per ulteriori informazioni, consulta the section called "Ripristina la versione di Kubernetes". Prima di eseguire l'aggiornamento a una nuova versione di Kubernetes, consigliamo di esaminare le informazioni contenute in Comprensione del ciclo di vita della versione di Kubernetes su EKS e nei passaggi di aggiornamento in questo argomento.

Le nuove versioni di Kubernetes hanno introdotto modifiche significative. Pertanto ti consigliamo di verificare il comportamento delle applicazioni rispetto alla nuova versione di Kubernetes prima di eseguire l'aggiornamento sui cluster di produzione. È possibile eseguire questa operazione mediante la creazione di un flusso di lavoro di integrazione continua per testare il comportamento totale dell'applicazione prima di passare a una nuova versione di Kubernetes.

Il processo di aggiornamento consiste nell'avvio, da parte di Amazon EKS, di nuovi nodi del server API con la versione aggiornata di Kubernetes per sostituire quelli esistenti. Amazon EKS esegue i controlli dell'integrità dell'infrastruttura standard e dello stato di preparazione per il traffico di rete su questi nuovi nodi per verificare che funzionino come previsto. Tuttavia, una volta avviato l'aggiornamento del cluster, non è possibile metterlo in pausa o interromperlo. Se uno di questi controlli non va a buon fine, Amazon EKS ripristina l'implementazione dell'infrastruttura e il cluster rimane nella versione precedente di Kubernetes. Le applicazioni in esecuzione non sono interessate

e il cluster non è mai lasciato in uno stato non deterministico o irrecuperabile. Amazon EKS esegue regolarmente il backup di tutti i cluster gestiti e, se necessario, dispone di meccanismi per il recupero dei cluster. Stiamo valutando e migliorando costantemente i nostri processi di gestione dell'infrastruttura di Kubernetes.

Per aggiornare il cluster, Amazon EKS richiede fino a cinque indirizzi IP disponibili dalle sottoreti specificate al momento della creazione del cluster. Amazon EKS crea nuove interfacce di rete elastiche (interfacce di rete) per il cluster in una delle sottoreti specificate, che può essere diversa rispetto a quella in cui si trovano le interfacce di rete esistenti. Assicurati quindi che le regole del gruppo di sicurezza consentano la comunicazione necessaria con il cluster per una delle sottoreti specificate al momento della creazione del cluster. Se una delle sottoreti specificate al momento della creazione del cluster non esiste, non dispone di un numero sufficiente di indirizzi IP disponibili o non dispone di regole per i gruppi di sicurezza che consentano le comunicazioni necessarie tra i cluster, l'aggiornamento può avere esito negativo.

Per garantire che l'endpoint del server API per il tuo cluster sia sempre accessibile, Amazon EKS fornisce un'elevata disponibilità del piano di controllo Kubernetes ed esegue aggiornamenti continui delle istanze del server API durante le operazioni di aggiornamento. Per tenere conto della modifica degli indirizzi IP delle istanze del server API che supportano l'endpoint del server API Kubernetes, devi assicurarti che i client del tuo server API gestiscano le riconnessioni in modo efficace. Le versioni recenti di `kubectl` e le librerie di client Kubernetes sono ufficialmente supportati, esegui questo processo di riconnessione in modo trasparente.

> **Note**
>
> Per ulteriori informazioni sui componenti di un aggiornamento del cluster, consulta Best practice per gli aggiornamenti dei cluster nella Guida alle best practice EKS. Questa risorsa consente di pianificare un aggiornamento e comprendere la strategia di aggiornamento di un cluster.

## Considerazioni per la modalità automatica di Amazon EKS

- La funzionalità di elaborazione della modalità automatica di Amazon EKS controlla la versione di Kubernetes dei nodi. Dopo aver aggiornato il piano di controllo, modalità automatica EKS inizierà ad aggiornare in modo incrementale i nodi gestiti. Modalità automatica EKS rispetta i budget relativi alle interruzioni dei pod.

- Non è necessario aggiornare manualmente le funzionalità di modalità automatica di Amazon EKS, incluse le funzionalità di scalabilità automatica di calcolo, archiviazione a blocchi e bilanciamento del carico.

> **Note**
>
> Se riscontri problemi dopo l'aggiornamento, puoi avviare un rollback. Per i cluster che eseguono la modalità automatica EKS, EKS ripristina automaticamente i nodi di lavoro prima di ripristinare il piano di controllo. Per ulteriori informazioni, consulta the section called "Ripristina la versione di Kubernetes".

## Riepilogo

Il riepilogo approfondito del processo di aggiornamento del cluster Amazon EKS è il seguente:

1. Assicurati che il cluster sia in uno stato che supporti un aggiornamento. Ciò include il controllo delle API Kubernetes usate dalle risorse distribuite nel cluster, per garantire che il cluster sia privo di problemi di salute. È necessario utilizzare gli approfondimenti sull'aggiornamento di Amazon EKS per valutare la preparazione all'aggiornamento del cluster.
2. Aggiornate il piano di controllo alla versione secondaria successiva (ad esempio, dalla 1.35 alla 1.36).
3. Aggiorna i nodi nel piano dati in modo che corrispondano a quelli del piano di controllo.
4. Aggiorna tutte le applicazioni aggiuntive eseguite sul cluster (ad esempio, `cluster-autoscaler`).
5. Esegui l'aggiornamento dei componenti aggiuntivi forniti da Amazon EKS, come quelli inclusi per impostazione predefinita:
   - Versione consigliata di Amazon VPC CNI
   - Versione consigliata di CoredNS
   - Versione suggerita `kube-proxy`
6. Aggiorna tutti i client che comunicano con il cluster (ad esempio, `kubectl`).

## Passaggio 1: preparazione per l'aggiornamento

Confrontare la versione Kubernetes del piano di controllo cluster con la versione Kubernetes dei nodi.

- Ottieni la versione di Kubernetes del piano di controllo del cluster.

```
kubectl version
```

- Ottieni la versione di Kubernetes dei tuoi nodi. Questo comando restituisce tutti i nodi gestiti e autogestiti Amazon EC2, Fargate e ibridi. Ciascun pod Fargate è presente nell'elenco assieme al rispettivo nodo.

```
kubectl get nodes
```

Prima di aggiornare il piano di controllo a una nuova versione di Kubernetes, assicurarsi che la versione secondaria di Kubernetes dei nodi gestiti e dei nodi Fargate nel cluster sia la stessa versione del piano di controllo. Ad esempio, se il piano di controllo è in versione running `1.29` e uno dei nodi è in esecuzione`1.28`, è necessario aggiornare i nodi alla versione `1.29` prima di aggiornare il control plane alla 1.30. Prima di aggiornare il piano di controllo, si consiglia inoltre di aggiornare i nodi autogestiti e i nodi ibridi alla stessa versione del piano di controllo. Per ulteriori informazioni, consultare the section called "Aggiorna", the section called "Metodi di aggiornamento" e the section called "Aggiornamento dei nodi ibridi". Se hai nodi Fargate con una versione secondaria inferiore rispetto alla versione del piano di controllo, devi prima eliminare il pod rappresentato dal nodo. a aggiornare il piano di controllo (control-plane). Tutti i pod rimanenti saranno aggiornati alla nuova versione dopo averli nuovamente implementati.

## Passaggio 2: esaminare le considerazioni sull'aggiornamento

Gli approfondimento sull'aggiornamento di Amazon EKS esegue automaticamente la scansione dei cluster e li mette a confronto con un elenco di potenziali problemi di aggiornamento della versione di Kubernetes, come l'utilizzo di API Kubernetes obsoleti. Amazon EKS aggiorna periodicamente l'elenco dei controlli approfonditi da eseguire in base alle valutazioni delle modifiche nel progetto Kubernetes. Amazon EKS aggiorna anche l'elenco dei controlli approfonditi man mano che sono introdotte modifiche nel servizio Amazon EKS insieme a nuove versioni. Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

Esamina la Guida alla migrazione delle API obsolete nei documenti di Kubernetes.

## Esaminare gli approfondimenti sull'aggiornamento

Utilizza approfondimenti sull'aggiornamento di Amazon EKS per identificare i problemi. Per ulteriori informazioni, consulta the section called "Visualizzare gli approfondimenti sugli aggiornamenti (console)".

## Considerazioni dettagliate

- Poiché Amazon EKS viene eseguito in un piano di controllo ad alta disponibilità, è possibile aggiornare solo una versione secondaria alla volta. Per ulteriori informazioni su questo requisito, consulta Kubernetes Version and Version Skew Support Policy (Policy per la versione Kubernetes e il supporto Skew della versione). Supponiamo che la versione corrente del cluster sia `1.28` e che desideri aggiornarlo alla versione `1.30`. Devi prima aggiornare la versione `1.28` del cluster alla versione `1.29`, quindi aggiornare la versione `1.29` del cluster alla versione `1.30`.
- Esamina la discrepanza di versione tra Kubernetes `kube-apiserver` e `kubelet` sui nodi.
  - A partire dalla versione `1.28` di Kubernetes, `kubelet` può essere fino a tre versioni minori precedenti rispetto a `kube-apiserver`. Consulta la sezione Kubernetes upstream version skew policy.
  - Se `kubelet` sui nodi gestiti e Fargate è nella versione di Kubernetes `1.25` o più recente, è possibile aggiornare il cluster fino a tre versioni successive senza aggiornare la versione di `kubelet`. Ad esempio, se `kubelet` è nella versione `1.25`, è possibile aggiornare la versione del cluster Amazon EKS dalla `1.25` alla `1.26`, `1.27` e `1.28` mentre `kubelet` rimane sulla versione `1.25`.
- Come best practice prima di iniziare un aggiornamento, assicurati che `kubelet` sui nodi sia alla stessa versione di Kubernetes del tuo piano di controllo.
- Se il cluster è configurato con una versione del plugin CNI di Amazon VPC per Kubernetes precedente a `1.8.0`, consigliamo di aggiornare il plugin all'ultima versione prima di aggiornare il cluster. Per aggiornare il plug-in, consulta the section called "CNI di Amazon VPC".
- Puoi eseguire un backup del tuo cluster Amazon EKS, per consentirti di ripristinare lo stato del cluster Amazon EKS e lo storage persistente in caso di errori durante il processo di aggiornamento. Per informazioni, consultare the section called " AWS Backup".

# Fase 3: aggiornamento del piano di controllo del cluster

> **Important**
>
> Amazon EKS ha temporaneamente eseguito il rollback di una funzionalità che richiedeva l'utilizzo di un flag `--force` per aggiornare il cluster in caso di determinati problemi di approfondimenti sul cluster. Per ulteriori informazioni, consulta la sezione Ripristino temporaneo dell'applicazione degli approfondimenti sull'aggiornamento della versione del cluster su. GitHub
> Amazon EKS aggiorna un approfondimento sul cluster 24 ore dopo "l'ultimo aggiornamento". È possibile confrontare l'ora in cui hai risolto un problema con "l'ora dell'ultimo aggiornamento" dell'approfondimento sul cluster.
> Inoltre, l'aggiornamento dello stato degli approfondimenti può richiedere fino a 30 giorni dopo la risoluzione dell'utilizzo di API obsolete. Gli approfondimenti sull'aggiornamento cercano sempre l'utilizzo di API obsolete in una finestra di 30 giorni consecutivi.

È possibile inviare la richiesta di aggiornamento della versione del tuo piano di controllo EKS utilizzando:

- eksctl
- la console AWS
- la AWS CLI

## Aggiornare il cluster: eksctl

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione e sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

Aggiorna la versione di Kubernetes del tuo piano di controllo di Amazon EKS. Sostituisci `<cluster-name>` con il nome del cluster. Sostituisci `<version-number>` con il numero di versione supportato

da Amazon EKS a cui aggiornare il tuo cluster. Per l'elenco completo delle versioni supportate, consulta Versioni supportate da Amazon EKS.

```
eksctl upgrade cluster --name <cluster-name> --version <version-number> --approve
```

Il processo di aggiornamento può richiedere alcuni minuti per il completamento.

Continua su the section called "Fase 4: aggiornamento dei componenti del cluster".

### Aggiorna cluster - AWS console

1. Aprire la Console Amazon EKS.
2. Scegli Aggiorna ora per un cluster che desideri aggiornare.
3. Seleziona la versione a cui aggiornare il cluster e scegli Aggiorna.
4. Il processo di aggiornamento può richiedere alcuni minuti per il completamento. Continua su the section called "Fase 4: aggiornamento dei componenti del cluster".

### Aggiorna cluster - AWS CLI

1. Verifica che la AWS CLI sia installata e che tu abbia effettuato l'accesso. Per ulteriori informazioni, consulta Installazione o aggiornamento alla versione più recente della AWS CLI.
2. Aggiorna il tuo cluster Amazon EKS con il seguente AWS comando CLI. Sostituisci `<cluster-name>` e `<region-code>` del cluster che desideri aggiornare. Sostituisci `<version-number>` con il numero di versione supportato da Amazon EKS a cui aggiornare il tuo cluster. Per l'elenco completo delle versioni supportate, consulta Versioni supportate da Amazon EKS.

   ```
   aws eks update-cluster-version --name <cluster-name> \
     --kubernetes-version <version-number> --region <region-code>
   ```

   Di seguito viene riportato un output di esempio.

   ```
   {
       "update": {
           "id": "<update-id>",
           "status": "InProgress",
           "type": "VersionUpdate",
           "params": [
               {
   ```

```
                "type": "Version",
                "value": "<version-number>"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
```

3. Il processo di aggiornamento può richiedere alcuni minuti per il completamento. É possibile monitorare lo stato di aggiornamento del cluster attraverso il seguente comando. Oltre a utilizzare gli stessi `<cluster-name>` e `<region-code>`, utilizza `<update-id>` restituito dal comando precedente.

```
aws eks describe-update --name <cluster-name> \
   --region <region-code> --update-id <update-id>
```

   Quando viene visualizzato lo stato `Successful`, l'aggiornamento è completo.

4. Continua su the section called "Fase 4: aggiornamento dei componenti del cluster".

## Fase 4: aggiornamento dei componenti del cluster

1. Dopo che l'aggiornamento del cluster è completo, aggiornare i nodi di lavoro alla stessa versione secondaria di Kubernetes del cluster aggiornato. Per ulteriori informazioni, consultare the section called "Metodi di aggiornamento", the section called "Aggiorna" e the section called "Aggiornamento dei nodi ibridi". Tutti i nuovi pod avviati in Fargate hanno una versione di `kubelet` che corrisponde alla versione del cluster. I pod Fargate esistenti non sono modificati.
2. (Facoltativo) Se è stato implementato Kubernetes Cluster Autoscaler nel cluster prima di aggiornarlo, aggiornare Cluster Autoscaler alla versione più recente che corrisponde alla versione principale e secondaria di Kubernetes per cui è stato eseguito l'aggiornamento.
   a. Apri la pagina delle versioni di Cluster Autoscaler in un browser Web e trova la versione più recente di Cluster Autoscaler corrispondente alla versione principale e secondaria di Kubernetes del cluster. Ad esempio, se la versione Kubernetes del cluster è `1.30`, cerca la versione più recente di Cluster Autoscaler che inizia con `1.30`. Registra il numero di versione semantica (`1.30.n`, ad esempio) per tale versione da utilizzare nella fase successiva.

b. Impostare il tag image di Cluster Autoscaler sulla versione registrata nella fase precedente mediante il comando seguente. Se necessario, sostituire `X.XX.X` con il valore in proprio possesso.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:vX.XX.X
```

3. (Solo cluster con nodi GPU) Se il tuo cluster ha gruppi di nodi con supporto GPU (ad esempio,`p3.2xlarge`), devi aggiornare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo `<vX.X.X>` con la versione desiderata prima di eseguire il comando seguente. NVIDIA/k8s-device-plugin

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<vX.X.X>/
deployments/static/nvidia-device-plugin.yml
```

4. Aggiorna il plugin CNI di Amazon VPC per Kubernetes, CoredNS e componenti aggiuntivi `kube-proxy`. È preferibile aggiornare i componenti aggiuntivi alle versioni minime elencate nei Token dell'account di servizio.
   - Se stai utilizzando i componenti aggiuntivi di Amazon EKS, nella console Amazon EKS seleziona Clusters (Cluster), quindi seleziona il nome del cluster aggiornato nel riquadro di navigazione a sinistra. Le notifiche vengono visualizzate nella console Ti viene segnalato che è disponibile una nuova versione per ogni componente aggiuntivo per cui è disponibile un aggiornamento. Per aggiornare un componente aggiuntivo, seleziona la Add-onsscheda. In una delle caselle relative al componente aggiuntivo che dispone di un aggiornamento disponibile, selezionare Aggiorna ora, selezionare una versione disponibile e quindi selezionare Aggiorna.
   - In alternativa, puoi utilizzare la AWS CLI `eksctl` o aggiornare i componenti aggiuntivi. Per ulteriori informazioni, consulta the section called “Aggiornamento di un componente aggiuntivo”.
5. Se necessario, aggiorna la tua versione di `kubectl`. Utilizza la versione secondaria `kubectl` immediatamente precedente a quella del piano di controllo del cluster Amazon EKS.

## Effettuare il downgrade della versione di Kubernetes di un cluster Amazon EKS

Puoi ripristinare la versione Kubernetes di un cluster Amazon EKS alla versione secondaria precedente entro 7 giorni dall'aggiornamento. Per ulteriori informazioni, consulta the section called “Ripristina la versione di Kubernetes”. Dopo la finestra di 7 giorni, non è possibile effettuare il

downgrade. Invece, crea un nuovo cluster su una versione precedente di Amazon EKS e migra i carichi di lavoro.

# Ripristina un cluster a una versione precedente di Kubernetes

Con il rollback della versione di Amazon EKS, puoi ripristinare il piano di controllo Kubernetes del cluster alla versione secondaria precedente dopo aver eseguito un aggiornamento sul posto. Se riscontri problemi dopo l'aggiornamento, come incompatibilità delle applicazioni, utilizzo obsoleto delle API o comportamenti imprevisti, puoi eseguire il rollback per ripristinare il cluster a un buono stato noto.

Durante un rollback, Amazon EKS ripristina il server API Kubernetes e i componenti del piano di controllo alla versione precedente preservando tutti i dati etcd, i carichi di lavoro dei clienti e i volumi persistenti.

## Cosa viene ripristinato

Viene eseguito il rollback dei seguenti componenti:

- Versione del server API Kubernetes
- I componenti del piano di controllo e le relative configurazioni
- Versione della piattaforma (torna alla versione più recente della piattaforma per la precedente versione di Kubernetes)
- Nodi di lavoro EKS Auto Mode.  Per i cluster che eseguono EKS Auto Mode, Amazon EKS gestisce automaticamente il rollback dei nodi di lavoro Auto Mode prima di ripristinare il piano di controllo. Per ulteriori informazioni, consulta the section called “Ripristina i cluster EKS Auto Mode”.

## Cosa NON viene ripristinato

I seguenti componenti non vengono ripristinati:

- dati  etcd. Tutto lo stato, le risorse e le configurazioni del cluster vengono preservati.
- Carichi di lavoro  dei clienti. I pod, le implementazioni e i servizi continuano a funzionare.
- Componenti aggiuntivi EKS.  Add-on le versioni rimangono invariate. Li gestisci separatamente.
- Volumi e dati persistenti. Tutti i dati dei clienti rimangono intatti.
- Self-managed nodi e nodi  ibridi. Sei responsabile del ripristino di questi dati.

- Gruppi di nodi gestiti. È necessario ripristinarli separatamente utilizzando l'`UpdateNodegroupVersion`API.

## Prerequisiti

Prima di poter ripristinare un cluster, devono essere soddisfatte tutte le seguenti condizioni:

| Requisito | Informazioni |
| --- | --- |
| Finestra di 7 giorni | È necessario avviare il rollback entro 7 giorni dal completamento dell'aggiornamento. Dopo 7 giorni, il rollback non è più disponibi le. |
| Cluster aggiornato | Il cluster deve essere stato aggiornato alla versione corrente tramite un aggiornamento sul posto. I cluster creati nella versione corrente non possono essere ripristinati. |
| Solo versione singola | È possibile eseguire il rollback solo di una versione secondari a (da N a N-1). Se hai eseguito l'aggiornamento dalla 1.31 alla 1.32 e poi alla 1.33, puoi ripristinare solo la versione 1.32, non la 1.31. |
| Versione supportata | Il rollback della versione è disponibile per le versioni Amazon EKS attualmente supportate. |
| Politica di supporto estesa | Per tornare a una versione con supporto esteso, devi prima modificare la politica di aggiornamento del cluster in`EXTENDED`. |
| Nessun aggiornamento automatico alla fine del supporto esteso | Se il cluster è stato aggiornato automaticamente al termine del supporto esteso, non puoi tornare alla versione precedent e. Se il cluster è stato aggiornato automaticamente al termine del supporto standard, puoi eseguire il rollback ma devi prima modificare la politica di aggiornamento in. `EXTENDED` |
| Stato del cluster | Lo `ACTIVE` stato del cluster deve essere attivo. Non è possibile avviare un rollback mentre è in corso un altro aggiornamento. |
| Compatibilità delle funzionalità EKS | Se una funzionalità EKS abilitata sul cluster non è supportat a nella versione precedente, la richiesta di rollback ha esito |

| Requisito | Informazioni |
| --- | --- |
| | negativo. Questo controllo non può essere aggirato con. `--force` |

Oltre ai requisiti precedenti, alcune condizioni rendono impossibile il rollback anche con la bandiera. `--force` Queste condizioni includono quanto segue: il cluster è stato creato nella versione corrente, sono trascorsi più di 7 giorni dall'aggiornamento, il cluster è già stato aggiornato nuovamente a una versione più recente o una funzionalità EKS incompatibile con le versioni precedenti è stata abilitata al limite della versione corrente.

## Riepilogo

Di seguito è riportato il riepilogo di alto livello del processo di rollback del cluster Amazon EKS:

1. Rivedi le informazioni sulla preparazione al rollback per identificare eventuali problemi che potrebbero influire sul rollback.
2. Risolvi eventuali problemi di blocco (informazioni sullo stato dell'ERRORE) o usali `--force` per aggirare i controlli approfonditi.
3. Verifica che le tue applicazioni, i controller personalizzati e gli strumenti di terze parti siano compatibili con la versione precedente di Kubernetes.
4. Se i tuoi nodi di lavoro eseguono la stessa versione di Kubernetes del piano di controllo, ripristina prima i nodi di lavoro.
5. Se disponi di componenti aggiuntivi che eseguono versioni incompatibili con la versione precedente di Kubernetes, esegui il downgrade a una versione compatibile.
6. Avvia il rollback del piano di controllo.
7. Monitora l'avanzamento del rollback.

> **Important**
>
> Per i cluster che eseguono EKS Auto Mode, il passaggio 4 viene gestito automaticamente. Quando avvii il rollback, Amazon EKS ripristina i nodi Auto Mode prima del piano di controllo. Per ulteriori informazioni, consulta the section called “Ripristina i cluster EKS Auto Mode”.

# Fase 1: Rivedi le informazioni sulla preparazione al rollback

Amazon EKS valuta automaticamente il cluster in base a una serie di controlli temporali di idoneità al rollback e individua eventuali problemi tramite le informazioni sui cluster incluse nella categoria. `ROLLBACK_READINESS` Queste informazioni vengono visualizzate dopo aver eseguito un aggiornamento e rimangono disponibili durante il periodo di idoneità al rollback di 7 giorni.

## Visualizzazione degli approfondimenti sulla preparazione al rollback

AWS Console:

1. Aprire la Console Amazon EKS.
2. Selezionare il cluster.
3. Scegli la scheda Approfondimenti sugli aggiornamenti. Le informazioni sulla disponibilità al rollback vengono visualizzate qui dopo un aggiornamento.
4. Esamina eventuali approfondimenti con stato ERRORE o AVVISO.

AWS CLI:

```
aws eks list-insights \
  --cluster-name my-cluster \
  --region us-west-2 \
  --filter '{"categories": ["ROLLBACK_READINESS"]}'
```

Per ottenere dettagli su una visione specifica:

```
aws eks describe-insight \
  --cluster-name my-cluster \
  --region us-west-2 \
  --id <insight-id>
```

## Approfondimenti rinfrescanti

Amazon EKS aggiorna le informazioni ogni 24 ore. Puoi attivare manualmente un aggiornamento dopo aver risolto i problemi scegliendo il  pulsante  Aggiorna nella console Amazon EKS o utilizzando l'interfaccia a riga di comando:

```
aws eks start-insights-refresh \
```

```
  --cluster-name my-cluster \
  --region us-west-2
```

> **Note**
>
> Amazon EKS aggiorna automaticamente le informazioni quando si avvia un rollback per assicurarsi che i controlli vengano eseguiti in base allo stato più recente del cluster.

## Comportamento relativo allo stato di

La tabella seguente descrive il significato di ogni stato di Insight e il relativo effetto sul rollback:

| Stato | Significato | Effetto sul rollback |
|---|---|---|
| PASSAGGIO | Nessun problema rilevato per questo controllo | Rollback consentito |
| ATTENZIONE | Potenziale problema rilevato, non bloccante | Rollback consentito (solo avviso) |
| ERROR (ERRORE) | È stato rilevato un problema di blocco | Rollback bloccato fino alla risoluzione o utilizzato `--force` per bypassare |
| UNKNOWN | Impossibile determinare lo stato | Rollback bloccato fino alla risoluzione o utilizzato `--force` per bypassare |

Gli approfondimenti con stato ERROR o UNKNOWN bloccano il rollback. Gli approfondimenti con stato PASSING o WARNING non impediscono il rollback.

## Controlli di idoneità al rollback

Amazon EKS esegue una serie di controlli nell'ambito delle informazioni sulla preparazione al rollback. Questi controlli valutano la compatibilità dell'utilizzo delle API (incluso il rilevamento delle modifiche a livello di campo), lo stato del cluster, l'inclinazione della versione di Kubelet, la distorsione della versione kube-proxy e la compatibilità delle versioni aggiuntive. Per i cluster che eseguono EKS Auto Mode, controlli aggiuntivi valutano i budget relativi alle interruzioni, le annotazioni relative alle interruzioni e le configurazioni. NodePool PodDisruptionBudget

### Usando il flag --force

Se le informazioni sulla preparazione al rollback mostrano lo stato ERROR e desideri procedere senza risolvere i problemi, puoi utilizzare il `--force` flag per bypassare tutti i controlli di approfondimento:

```
aws eks update-cluster-version \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --force \
  --region us-west-2
```

> **Warning**
>
> L'utilizzo `--force` ignora tutti i controlli approfonditi (ERROR, WARNING, UNKNOWN) e procede direttamente con il rollback. Amazon EKS non può garantire la sicurezza del rollback quando i controlli approfonditi vengono ignorati. Ti assumi la piena responsabilità per eventuali problemi che dovessero insorgere.

La `--force` bandiera ignora solo i controlli approfonditi. Non ignora le convalide dei prerequisiti come la finestra di 7 giorni, il controllo della versione di creazione o il controllo di rollback sequenziale. Per i cluster in modalità automatica, non sovrascrive i controlli sulle interruzioni. `--force` NodePool i budget relativi alle interruzioni, i PDB e le annotazioni relative al divieto di interruzione delle attività vengono comunque rispettati.

## Fase 2: Preparazione dei nodi di lavoro

Prima di ripristinare il piano di controllo, assicurati che i nodi di lavoro siano compatibili con la versione di destinazione. La policy di distorsione delle versioni di Kubernetes richiede che i nodi di lavoro non possano eseguire una versione più recente del piano di controllo.

### Modalità automatica di EKS

Nessuna operazione necessaria. Quando avvii il rollback, Amazon EKS ripristina automaticamente i nodi Auto Mode prima del piano di controllo. Per ulteriori informazioni, consulta the section called "Ripristina i cluster EKS Auto Mode".

### Gruppi di nodi gestiti (MNG)

È necessario ripristinare i gruppi di nodi gestiti alla versione precedente prima di ripristinare il piano di controllo. Usa l'`UpdateNodegroupVersion`API:

```
aws eks update-nodegroup-version \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --kubernetes-version 1.30 \
  --region us-west-2
```

L'aggiornamento del gruppo di nodi rispetta le impostazioni di aggiornamento configurate (`maxUnavailable`o`maxUnavailablePercentage`) e la strategia di aggiornamento (Rolling o Force).

### Self-managed nodi e nodi ibridi

Sei responsabile del ripristino dei nodi autogestiti e dei nodi ibridi. Aggiorna le AMI o le configurazioni dei nodi per utilizzare la versione precedente di Kubernetes prima di ripristinare il piano di controllo.

### Fargate

Il rollback della versione non è supportato per i nodi di lavoro Fargate. È possibile ripristinare il piano di controllo di un cluster che utilizza Fargate, ma i pod Fargate che eseguono la stessa versione di Kubernetes del piano di controllo attivano lo skew insight della versione kubelet con lo stato ERROR.

Amazon EKS non è in grado di ripristinare automaticamente i pod Fargate a una versione precedente di Kubelet.

Soluzione alternativa:  se hai pod Fargate che eseguono la stessa versione di Kubernetes del piano di controllo, eliminali prima di avviare il rollback. Quindi ripristina il piano di controllo. Tutti i pod rimanenti vengono avviati con la versione ripristinata quando li ridistribuisci.

In alternativa, usala `--force` per bypassare l'insight check. Tuttavia, una violazione della distorsione della versione di Kubelet potrebbe causare un comportamento imprevisto dei carichi di lavoro di Fargate fino alla sostituzione dei pod.

## Fase 3: Ripristina il piano di controllo del cluster

È possibile avviare un rollback utilizzando la AWS console, la AWS CLI o l'API EKS.

## Eseguire il rollback di un cluster utilizzando AWS Console

1. Aprire la Console Amazon EKS.
2. Selezionare il cluster.
3. Scegli il menu a  discesa  Azioni.
4. Scegli la versione del cluster  Rollback.
5. Consulta il riepilogo del rollback, inclusi eventuali avvisi di approfondimento.
6. Scegli la versione  Rollback.

Il completamento del rollback richiede alcuni minuti. Per i cluster in modalità automatica, la fase di rollback del nodo potrebbe richiedere più tempo. Per ulteriori informazioni, consulta the section called "Ripristina i cluster EKS Auto Mode".

## Esegui il rollback di un cluster utilizzando AWS CLI

Usa il `update-cluster-version` comando esistente con la versione precedente (N-1) di Kubernetes:

```
aws eks update-cluster-version \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --region us-west-2
```

Risposta di esempio:

```
{
    "update": {
        "id": "e4091a28-ea14-48fd-a8c7-975aeb469e8a",
        "status": "InProgress",
        "type": "VersionRollback",
        "params": [
            {
                "type": "Version",
                "value": "1.30"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.16"
```

```
            }
        ],
        "createdAt": "2026-05-12T16:56:01.082000-04:00",
        "errors": []
    }
}
```

> **Note**
>
> Amazon EKS esegue un aggiornamento delle informazioni prima di eseguire il rollback se i dati di insight sono obsoleti.

## Fase 4: Monitora l'avanzamento del rollback

Puoi monitorare lo stato del rollback del cluster utilizzando la console Amazon EKS o l' AWS interfaccia a riga di comando.

AWS CLI:

```
aws eks describe-update \
  --name my-cluster \
  --region us-west-2 \
  --update-id e4091a28-ea14-48fd-a8c7-975aeb469e8a
```

AWS Console:

1. Aprire la Console Amazon EKS.
2. Selezionare il cluster.
3. Scegli la  scheda Cronologia  aggiornamenti.
4. Individua l'ID di aggiornamento associato al rollback per visualizzarne lo stato attuale.

### Transizioni di stato

Per i cluster standard (senza modalità automatica):

```
InProgress # Successful
```

```
InProgress # Failed
```

Per i cluster in modalità automatica, lo stato del cluster rimane attivo `ACTIVE` durante il rollback dei nodi e cambia `UPDATING` solo quando inizia il rollback del piano di controllo. `describe-update`Da utilizzare per tenere traccia dell'avanzamento complessivo del rollback. Per ulteriori informazioni, consulta the section called "Ripristina i cluster EKS Auto Mode".

Quando viene visualizzato uno `Successful` stato, il rollback è completo.

## Considerazioni e avvertenze

### Gli approfondimenti sono la cosa migliore da fare e sono puntuali

Le informazioni sui cluster vengono valutate nel momento in cui viene attivato il rollback. Se apporti modifiche al cluster dopo aver controllato gli insight ma prima del completamento del rollback (ad esempio, creando risorse utilizzando nuove API), tali modifiche non vengono rilevate dal controllo di approfondimento iniziale e potrebbero causare problemi una volta completato il rollback.

### conservazione dei dati etcd

Amazon EKS conserva i dati etcd durante il rollback. Le risorse incompatibili bypassate utilizzando il `--force` flag rimangono persistenti e non vengono raccolte inutili.

### Costi di assistenza estesa

Se si esegue il rollback da una versione con supporto standard a una versione con supporto esteso, il cluster inizia a incorrere in costi di supporto esteso. Ad esempio, se esegui l'upgrade da 1.30 (supporto esteso) a 1.31 (supporto standard) e poi ritorni alla 1.30, i costi del supporto esteso riprendono.

### Modello di responsabilità condivisa per il rollback

Amazon EKS ripristina il piano di controllo di Kubernetes alla versione desiderata. Come parte del modello di responsabilità condivisa, sei responsabile della verifica della compatibilità delle applicazioni con la versione precedente:

- Amazon EKS è responsabile del ripristino sicuro dei componenti del piano di controllo.
- È tua responsabilità assicurarti che le tue applicazioni, configurazioni e dipendenze siano compatibili con la versione precedente.

- È necessario esaminare eventuali incompatibilità tra le versioni, valutare l'esposizione del cluster e mitigare eventuali problemi.

### CloudFormation comportamento di rollback dello stack

Se un aggiornamento CloudFormation dello stack AWS fallisce e attiva un rollback dello stack, il ripristino a una versione precedente del modello che specifica una versione precedente di Kubernetes non attiva il rollback della versione del cluster. Il rollback della versione deve essere avviato esplicitamente tramite l'API, la CLI o la console. `UpdateClusterVersion`

## Rollback e componenti aggiuntivi

Amazon EKS non esegue automaticamente il rollback delle versioni aggiuntive durante il rollback di una versione del cluster. È necessario gestire le versioni aggiuntive separatamente.

Prima di ripristinare il piano di controllo:

1. Verifica la compatibilità del componente aggiuntivo con la versione di destinazione utilizzando le informazioni sulla disponibilità al rollback.
2. Se una versione aggiuntiva è incompatibile con la versione precedente di Kubernetes, esegui prima il downgrade:

```
aws eks update-addon \
  --cluster-name my-cluster \
  --addon-name vpc-cni \
  --addon-version v1.22.4-eksbuild.3 \
  --region us-west-2
```

3. Una volta completato il rollback del piano di controllo, verifica che tutti i componenti aggiuntivi funzionino correttamente.

> **Note**
>
> Gli approfondimenti sulla preparazione al rollback controllano solo le versioni dei componenti aggiuntivi. EKS-managed Per i componenti aggiuntivi autogestiti, sei responsabile della convalida della compatibilità con la versione di destinazione prima del rollback.

## Risorse correlate

- the section called "Ripristina i cluster EKS Auto Mode"
- the section called "Aggiornamento della versione di Kubernetes"
- the section called "Approfondimenti sui cluster"
- Comprendi il ciclo di vita della versione di Kubernetes su Amazon EKS
- Aggiorna un gruppo di nodi gestiti
- Procedure consigliate per gli aggiornamenti dei cluster

# Cluster Rollback EKS Auto Mode

Quando avvii un rollback di versione su un cluster che esegue EKS Auto Mode, Amazon EKS gestisce automaticamente il rollback dei nodi di lavoro in modalità automatica prima di ripristinare il piano di controllo. Questa pagina spiega come funziona il rollback dei nodi in modalità automatica, come velocizzarlo e come annullarlo se necessario.

Per informazioni generali sul rollback della versione, inclusi i prerequisiti, i controlli approfonditi e il processo di rollback complessivo, consulta. the section called "Ripristina la versione di Kubernetes"

## Come funziona il rollback in modalità automatica

EKS Auto Mode utilizza un Karpenter-based sistema per gestire l'infrastruttura dei nodi di lavoro, inclusi gli aggiornamenti e i rollback delle versioni di Kubernetes. Quando chiamate `UpdateClusterVersion` con la versione precedente (N-1) su un cluster con la modalità automatica abilitata, EKS esegue la seguente sequenza:

1. Convalida i prerequisiti e aggiorna le informazioni sulla preparazione al rollback.
2. Indirizza i nodi verso la versione di rollback desiderata utilizzando un sistema, rispettando i controlli di interruzione configurati. Karpenter-based
3. Dopo che tutti i nodi rientrano nella policy di distorsione della versione di Kubernetes per la versione di rollback desiderata, EKS ricontrolla le informazioni e procede con il rollback del piano di controllo.

Il piano di controllo rimane sulla versione corrente (più recente) e continua a servire il traffico normalmente mentre i nodi eseguono il rollback. La policy di inclinazione della versione di Kubernetes

consente ai nodi di eseguire fino a tre versioni minori precedenti al kube-apiserver, quindi questo stato intermedio è valido.

> **Note**
>
> Il rollback viene attivato utilizzando la stessa API e lo stesso processo descritti in. the section called "Ripristina la versione di Kubernetes" Non esiste un'API separata per il rollback del nodo Auto Mode.

> **Note**
>
> La fase di rollback del nodo (fase 2) può richiedere da pochi minuti a 7 giorni, a seconda dei controlli di interruzione. Se il rollback del nodo non viene completato entro il timeout configurato, l'aggiornamento viene contrassegnato come non riuscito.

> **Note**
>
> Mentre è in corso un rollback, gli altri aggiornamenti del piano di controllo attivati dal cliente vengono bloccati. Per eseguire un aggiornamento diverso, annulla prima il rollback utilizzando l'API. CancelUpdate

## Stato del cluster durante il rollback

| Fase | Stato del cluster | Cosa sta succedendo |
|---|---|---|
| Ripristino del nodo in corso | ACTIVE | Karpenter sta sostituendo i nodi con l'AMI della versione precedente. Il piano di controllo è funzionante e serve il traffico nella versione attuale. |
| Control plane rollback | AGGIORNAMENTO IN CORSO | I componenti del server API e del piano di controllo vengono ripristinati alla versione precedente. |

| Fase | Stato del cluster | Cosa sta succedendo |
|---|---|---|
| Rollback completato | ACTIVE | Cluster è completamente aggiornato alla versione precedente. |

Lo stato del cluster rimane `ACTIVE` durante la fase di rollback del nodo. Usa `ListUpdates` o `DescribeUpdate` per determinare se è in corso un rollback. Nella console Amazon EKS, accedi al tuo cluster e apri la scheda Cronologia degli aggiornamenti per visualizzare lo stato dell'ID di aggiornamento associato al rollback.

Per monitorare l'avanzamento dei singoli nodi durante il rollback, controlla la versione Kubernetes dei tuoi nodi Auto Mode:

```
kubectl get nodes -l karpenter.sh/nodepool=<nodepool-name> -o wide
```

## Controlli delle interruzioni

Il rollback in modalità automatica rispetta tutti i controlli di interruzione esistenti. Questi controlli determinano la velocità con cui i nodi possono essere sostituiti e potrebbero influire in modo significativo sulla durata del rollback.

NodePool Budget per le interruzioni

NodePool i budget per le interruzioni controllano quanti nodi possono essere interrotti contemporaneamente. Durante il rollback, Karpenter rispetta questi budget quando riporta i nodi alla versione precedente.

- Un budget di quattro drift blocca il rollback a tempo indeterminato. `nodes: 0` Ciò attiva un'analisi ERROR.
- Un budget restrittivo (ad esempio`nodes: 1`) rallenta il rollback ma consente l'avanzamento.

Esempio NodePool con un budget per le interruzioni che consente di sostituire il 10% dei nodi alla volta:

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: default
```

```
spec:
  disruption:
    budgets:
      - nodes: "10%"
        reasons:
          - Drifted
```

Per ulteriori informazioni sulla configurazione dei budget per le NodePool interruzioni, consulta Create a Node Pool for EKS Auto Mode e Karpenter per i budget relativi alle interruzioni.

PodDisruptionBudgets

Kubernetes viene premiata durante la sostituzione dei nodi. PodDisruptionBudgets Se un PDB impedisce lo sfratto dei pod, l'interruzione del nodo viene ritardata fino al. TerminationGracePeriod

- PDB con interruzione del nodo di ritardo. `maxUnavailable: 0` Ciò attiva un'analisi di AVVISO.
- I PDB non bloccano in modo permanente il rollback ma possono rallentarlo in modo significativo.

Per ulteriori informazioni, consulta Proteggi i carichi di lavoro critici con un PDB e la documentazione PDB di Kubernetes.

Do-not-disrupt annotazioni

L'`karpenter.sh/do-not-disrupt`annotazione può essere impostata su nodi o pod:

Sui nodi: blocca l'interruzione dei nodi a tempo indeterminato. Ciò attiva un'analisi ERROR e deve essere rimosso prima che il rollback possa procedere su quel nodo.

Sui pod: ritarda l'interruzione del nodo fino al. TerminationGracePeriod Ciò attiva un'analisi di AVVISO ma non blocca in modo permanente il rollback.

Per ulteriori informazioni sul comportamento delle interruzioni di Karpenter, consulta la documentazione relativa alle interruzioni di Karpenter.

## Accelerazione del rollback

Se il rollback richiede più tempo del previsto, puoi modificare i controlli delle interruzioni mentre è in corso il rollback.

Aumenta i budget per le interruzioni NodePool

Modifica la NodePool risorsa per consentire più sostituzioni simultanee dei nodi:

```
kubectl edit nodepool default
```

Modifica il budget impostandolo su un valore più alto:

```
spec:
  disruption:
    budgets:
      - nodes: "50%"
        reasons:
          - Drifted
```

Rimuovi le annotazioni «non interrompere» dai nodi

Elenca i nodi con l'annotazione e rimuovila:

```
# List nodes with the annotation
kubectl get nodes -o json | jq '.items[] | select(.metadata.annotations["karpenter.sh/
do-not-disrupt"] == "true") | .metadata.name'

# Remove from a specific node
kubectl annotate node <node-name> karpenter.sh/do-not-disrupt-
```

Regola PodDisruptionBudgets

Se i PDB rallentano lo sfratto dei pod, modificali temporaneamente:

```
kubectl edit pdb <pdb-name> -n <namespace>
```

> **Warning**
>
> La regolazione dei controlli delle interruzioni influisce sulle garanzie di disponibilità delle applicazioni. Assicuratevi di comprenderne l'impatto prima di apportare modifiche alla produzione.

Per ulteriori informazioni sulla gestione dei controlli delle interruzioni, consulta Prevenzione dell'interruzione di pod e nodi in Amazon EKS Auto Mode.

## Annullamento di un rollback

Un rollback è in uno stato annullabile solo durante il rollback dei nodi. È possibile utilizzare CancelUpdate durante questa fase per interrompere l'operazione.

```
aws eks cancel-update \
  --name my-cluster \
  --update-id <update-id> \
  --region us-west-2
```

Annulla il comportamento

| Aspetto | Comportamento |
| --- | --- |
| Quando annullabile | Solo durante il rollback dei nodi della modalità Auto (prima che inizi il rollback del piano di controllo) |
| Semantica | Best-effort fermare. Interrompe l'operazione di rollback del nodo. |
| Mid-disruption nodi | Se un nodo è a metà dell'interruzione al momento dell'annu llamento, completa l'operazione corrente. |
| Post-cancellation stato | Aggiorna le transizioni da `Cancelling` a. `Cancelled` |
| Stato del cluster | Rimane `ACTIVE` dappertutto. |

Dopo la cancellazione

In caso di annullamento riuscito, i nodi passano alla versione corrente del cluster come di consueto. L'aggiornamento passa da a`Cancelling`. `Cancelled`

Dopo l'annullamento, puoi immediatamente:

- Riprova a eseguire il rollback (purché rientri nella finestra di idoneità di 7 giorni).
- Esegui un aggiornamento diverso del cluster.
- Lascia il cluster così com'è nella versione corrente.

Quando l'annullamento non è possibile

L'annullamento ha esito negativo se il rollback del nodo è già completo e il rollback del piano di controllo è iniziato o se l'aggiornamento è già stato completato con `Successful` o`Failed`.

> **Note**
>
> CloudFormation e Terraform non supportano direttamente l'API. CancelUpdate Se è necessario annullare un rollback avviato tramite IAc, è necessario chiamare direttamente l'API.

## Timeout di rollback

Il rollback del nodo Auto Mode ha un timeout configurabile controllato dal parametro in. `timeoutMinutes` `rollbackConfig` Il timeout predefinito è 720 minuti (12 ore). È possibile impostare un valore compreso tra 120 minuti (2 ore) e 10080 minuti (7 giorni). Il timeout è una proprietà con limite minimo, il che significa che non si verifica prima del tempo specificato, ma può verificarsi poco dopo.

```
aws eks update-cluster-version \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --rollback-config timeoutMinutes=1440 \
  --region us-west-2
```

Se tutti i nodi non hanno completato il rollback entro il timeout specificato:

1. Il timeback scade.
2. I nodi iniziano a tornare alla versione corrente del cluster.
3. Il piano di controllo rimane nella versione corrente (non è mai stato ripristinato).
4. Lo stato di aggiornamento passa a. `Failed`

Dopo un timeout, puoi riprovare a eseguire il rollback se sei ancora entro il periodo di validità del rollback di 7 giorni dall'aggiornamento originale. In pratica, se il rollback del nodo scade al giorno 7, è probabile che anche la finestra di idoneità al rollback sia scaduta, poiché entrambi sono di 7 giorni.

Per evitare i timeout, consulta le informazioni sulla preparazione al rollback prima di iniziare il rollback. Questi approfondimenti segnalano eventuali interruzioni, budget o annotazioni che potrebbero rallentare il processo di rollback dei nodi.

## Il flag --force e la modalità Auto

Il `--force` flag on ignora `UpdateClusterVersion` solo i controlli di analisi approfondita del cluster. Non ha alcun effetto sul comportamento di interruzione dei nodi in modalità automatica.

Anche con`--force`:

- NodePool i budget destinati alle interruzioni vengono comunque rispettati.
- PodDisruptionBudgets sono ancora onorati.
- Do-not-disrupt le annotazioni sono ancora rispettate.
- Il timeout di rollback del nodo di 7 giorni è ancora valido.

L'unico modo per accelerare il rollback dei nodi è modificare autonomamente i controlli di interruzione. Per informazioni dettagliate, vedi the section called “Accelerazione del rollback”.

## conflitti di timeout IAc

Infrastructure-as-Code gli strumenti presentano limitazioni di timeout che potrebbero entrare in conflitto con la durata del rollback in modalità automatica. CloudFormation consente fino a 36 ore per risorsa. Se l'operazione scade, la CloudFormation considera come un'operazione non operativa, il che può lasciare il cluster in uno stato diverso in cui il modello non riflette la versione effettiva del cluster. Il rollback della versione deve essere avviato in modo esplicito. Terraform Enterprise/Cloud ha un timeout di circa 24 ore, sebbene i timeout lato client possano variare a seconda della scadenza delle credenziali e di altri fattori.

Per allineare la durata del rollback con lo strumento IaC, utilizza il parametro in per impostare un timeout appropriato. `timeoutMinutes` `rollbackConfig` Se il tuo strumento IAc va in timeout, usa direttamente l' CancelUpdate API per riprendere il controllo. Se disponi di budget limitati per le interruzioni, valuta la possibilità di avviare il rollback direttamente tramite CLI o API anziché IAc.

### Aggiornamenti di sistema durante il rollback del nodo

Mentre i nodi della modalità automatica vengono ripristinati, EKS continua a mantenere il piano di controllo sicuro e disponibile.

Customer-triggered gli aggiornamenti (come UpdateClusterVersion o UpdateClusterConfig) vengono bloccati mentre è in corso il rollback del nodo. Se devi eseguire un aggiornamento ad alta priorità, annulla prima il rollback utilizzando`CancelUpdate`, quindi esegui l'aggiornamento e riavvia il rollback se è ancora all'interno della finestra di idoneità.

### Risorse correlate

- the section called "Ripristina la versione di Kubernetes"
- Panoramica della modalità automatica EKS
- Crea un pool di nodi per EKS Auto Mode
- Aggiorna la versione Kubernetes di un cluster EKS Auto Mode
- Prevenzione delle interruzioni di pod e nodi in Amazon EKS Auto Mode
- Risolvi i problemi relativi alla modalità EKS Auto
- Documentazione sulle interruzioni di Karpenter
- Kubernetes PodDisruptionBudgets

## Eliminazione di un cluster

Terminato l'utilizzo del cluster Amazon EKS, è necessario eliminare le risorse a esso associate per non dover sostenere costi superflui.

Puoi eliminare un cluster con, la o la CLI. `eksctl` Console di gestione AWS AWS

### Considerazioni

- Se viene visualizzato un errore in seguito alla rimozione del creatore del cluster, consultare questo articolo per la risoluzione.
- Le risorse del Servizio gestito da Amazon per Prometheus non rientrano nel ciclo di vita del cluster e devono essere gestite indipendentemente dal cluster. Quando si elimina il cluster, assicurarsi

di eliminare anche tutti gli scraper applicabili per bloccare i relativi costi. Per ulteriori informazioni, consultare Find and delete scrapers nella Guida per l'utente del Servizio gestito da Amazon per Prometheus.

- Per rimuovere un cluster connesso, consultare the section called "Annullamento della registrazione di un cluster"
- Prima di poter eliminare un cluster, assicurati che la protezione dall'eliminazione sia disabilitata per il cluster.
- Se il cluster dispone di funzionalità EKS (ACK, Argo CD o kro), è necessario eliminarle prima di poter eliminare il cluster. Per ulteriori informazioni, consulta the section called "Eliminare una funzionalità".

### Considerazioni per la modalità automatica EKS

- Eventuali nodi della modalità automatica EKS saranno eliminati, comprese le istanze gestite EC2.
- Tutti i bilanciatori del carico saranno eliminati.

Per ulteriori informazioni, consulta the section called "Disabilita modalità automatica di EKS".

## Prerequisiti

Di seguito sono riportati i passaggi da eseguire prima di poter eliminare un cluster. Questi passaggi si applicano indipendentemente dal metodo utilizzato per eliminare il cluster.

1. Elenca tutti i servizi in esecuzione nel cluster.

```
kubectl get svc --all-namespaces
```

2. Elimina i servizi che hanno un valore `EXTERNAL-IP` associato. Questi servizi sono anticipati da un Elastic Load Balancing, (load balancer) ed è necessario eliminarli in Kubernetes per consentire al sistema e alle risorse associate di essere rilasciate correttamente. Sostituisci *`service-name`* con il nome di ogni servizio elencato come descritto.

```
kubectl delete svc service-name
```

3. Elimina anche tutte le risorse di ingresso. Se non elimini le risorse in ingresso, l'application load balancer rimane attivo anche se hai eliminato il cluster. Sostituiscilo *`ingress-name`* con il nome delle tue risorse di ingresso.

```
kubectl get ingress --all-namespaces
```

```
kubectl delete ing ingress-name
```

## Eliminare un cluster (eksctl)

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

1. Segui i passaggi dei prerequisiti. Dopo averlo fatto, elimina il cluster e i nodi associati con il seguente comando, sostituendolo *`prod`* con il nome del cluster.

   ```
   eksctl delete cluster --name prod
   ```

   Output:

   ```
   [#]  using region region-code
   [#]  deleting EKS cluster "prod"
   [#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
   [#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
   [#]  will delete stack "eksctl-prod-cluster"
   [#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster. If in
    doubt, check CloudFormation console
   ```

## Elimina cluster (AWS console)

1. Segui i passaggi dei prerequisiti. Dopo averlo fatto, elimina tutti i gruppi di nodi, i profili Fargate e le funzionalità.

   a. Aprire la Console Amazon EKS.

b. Nel pannello di navigazione a sinistra, scegli Clusters (Cluster) Amazon EKS, quindi nell'elenco a schede dei cluster scegli il nome del cluster da eliminare.

c. Seleziona la scheda Compute (Calcolo), quindi scegli un gruppo di nodi da eliminare. Scegli Delete (Elimina), immetti il nome del gruppo di nodi, quindi seleziona Delete (Elimina). Eliminare tutti i gruppi di nodi del cluster.

> **Note**
>
> L'elenco presenta solo gruppi di nodi gestiti.

d. Scegli un profilo Fargate da eliminare, seleziona Delete (Elimina), immetti il nome del profilo e infine scegli Delete (Elimina). Eliminare tutti i profili di Fargate nel cluster.

e. Scegli la scheda Funzionalità e scegli una funzionalità da eliminare. Scegli Elimina funzionalità, inserisci il nome della funzionalità, quindi scegli Elimina. Elimina tutte le funzionalità nel cluster. Per ulteriori informazioni, consulta the section called "Eliminare una funzionalità".

2. Elimina tutti gli stack di nodi AWS CloudFormation autogestiti.

   a. Apri la AWS CloudFormation console.

   b. Scegli lo stack del nodo da eliminare, quindi scegli Elimina.

   c. Nella finestra di dialogo di conferma Delete stack (Elimina stack) scegliere Delete stack.(Elimina stack). Eliminare tutte le pile di nodi autogestiti nel cluster.

3. Eliminare il cluster.

   a. Aprire la Console Amazon EKS.

   b. Scegli il cluster da eliminare e scegli Elimina.

   c. Nella schermata di conferma dell'eliminazione del cluster, scegliere Elimina.

4. (Facoltativo) Elimina lo AWS CloudFormation stack VPC.

   a. Apri la AWS CloudFormation console.

   b. Selezionare lo stack del VPC da eliminare, quindi scegliere Delete (Elimina).

   c. Nella finestra di dialogo di conferma Delete stack (Elimina stack) scegliere Delete stack.(Elimina stack).

# Elimina cluster (AWS CLI)

1. Segui i passaggi dei prerequisiti. Dopo averlo fatto, elimina tutti i gruppi di nodi, i profili Fargate e le funzionalità.

   a. Elencare i gruppi di nodi nel cluster con il comando seguente.

   ```
   aws eks list-nodegroups --cluster-name my-cluster
   ```

   > **Note**
   >
   > L'elenco presenta solo gruppi di nodi gestiti.

   b. Eliminare ogni gruppo di nodi con il comando seguente. Eliminare tutti i gruppi di nodi del cluster.

   ```
   aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
   ```

   c. Elenca i proqfili Fargate nel cluster con il comando seguente.

   ```
   aws eks list-fargate-profiles --cluster-name my-cluster
   ```

   d. Eliminare ogni profilo di Fargate con il comando seguente. Eliminare tutti i profili di Fargate nel cluster.

   ```
   aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --
   cluster-name my-cluster
   ```

   e. Elenca le funzionalità del tuo cluster con il seguente comando.

   ```
   aws eks list-capabilities --cluster-name my-cluster
   ```

   f. Elimina ogni funzionalità con il seguente comando. Elimina tutte le funzionalità del cluster. Per ulteriori informazioni, consulta the section called "Eliminare una funzionalità".

   ```
   aws eks delete-capability --cluster-name my-cluster --capability-name my-
   capability-name
   ```

2. Elimina tutti gli stack di nodi AWS CloudFormation autogestiti.

a. Elenca gli AWS CloudFormation stack disponibili con il seguente comando. Trovare il nome del modello del nodo nell'output risultante.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. Elimina lo stack di ogni nodo con il seguente comando, sostituendo *node-stack* con il nome del tuo stack. Eliminare tutte le pile di nodi autogestiti nel cluster.

```
aws cloudformation delete-stack --stack-name node-stack
```

3. Elimina il cluster con il seguente comando, sostituendo *my-cluster* con il nome del tuo cluster.

```
aws eks delete-cluster --name my-cluster
```

4. (Facoltativo) Elimina lo stack VPC. AWS CloudFormation

a. Elenca gli AWS CloudFormation stack disponibili con il seguente comando. Trovare il nome del modello di VPC nell'output risultante.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. Elimina lo stack VPC con il seguente comando, sostituendo *my-vpc-stack* con il nome dello stack VPC.

```
aws cloudformation delete-stack --stack-name my-vpc-stack
```

## Proteggi i cluster EKS dall'eliminazione accidentale

L'eliminazione accidentale di un cluster EKS può compromettere le operazioni del cluster Kubernetes.

Ora è possibile proteggere i cluster EKS dall'eliminazione accidentale. Se la protezione dall'eliminazione è abilitata su un cluster, prima di poterlo eliminare è necessario disabilitare tale protezione.

Lo scopo della protezione dall'eliminazione è prevenire gli incidenti. È necessario limitare attentamente gli utenti autorizzati a eliminare i cluster.

Se tenti di eliminare un cluster attivo con la protezione dall'eliminazione attivata, riceverai un`InvalidRequestException`.

> **Important**
>
> Se abiliti la protezione da eliminazione su un cluster, devi disporre sia delle UpdateClusterConfig autorizzazioni che di DeleteCluster IAM per rimuovere prima la protezione dall'eliminazione e infine eliminare il cluster.

> **Note**
>
> Se lo stato del cluster è Creazione, Non riuscito o Eliminazione, è possibile eliminarlo anche se è attivata la protezione dall'eliminazione.

### Per abilitare la protezione dall'eliminazione per un cluster esistente

È possibile eseguire questo comando solo su un cluster con stato attivo.

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --deletion-protection
```

### Per disabilitare la protezione dall'eliminazione per un cluster esistente

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --no-deletion-protection
```

## Endpoint del server API del cluster

Questo argomento consente di abilitare l'accesso privato per l'endpoint del server API Kubernetes del cluster Amazon EKS e limitare, o disabilitare completamente, l'accesso pubblico in modo che non sia accessibile da Internet.

Quando si crea un nuovo cluster, Amazon EKS crea un endpoint per il server API Kubernetes gestito utilizzato per comunicare con il cluster (usando strumenti di gestione Kubernetes, ad esempio `kubectl`). Per impostazione predefinita, questo endpoint del server API è pubblico su Internet e l'accesso al server API è protetto utilizzando una combinazione di AWS Identity and Access Management (IAM) e RBAC (Kubernetes Role Based Access Control) nativo. Questo endpoint è noto come l'endpoint pubblico del cluster. Esiste anche un endpoint privato del cluster. Per maggiori

informazioni sull'endpoint privato del cluster, consulta la sezione the section called "Endpoint privato del cluster" seguente.

## `Formato endpoint del cluster` IPv6

EKS crea un endpoint dual-stack unico nel seguente formato per i nuovi cluster `IPv6` creati dopo ottobre 2024. Un cluster IPv6 è un cluster in cui si seleziona `IPv6` nell'impostazione della famiglia IP (`ipFamily`) del cluster.

Example

AWS

Endpoint del cluster EKS: public/private `eks-cluster.`*`region`*`.api.aws`

AWS GovCloud (US)

public/private Endpoint del cluster EKS: `eks-cluster.`*`region`*`.api.aws`

Amazon Web Services in China

public/private Endpoint del cluster EKS: `eks-cluster.`*`region`*`.api.amazonwebservices.com.cn`

> **Note**
>
> L'endpoint del cluster dual-stack è stato introdotto nell'ottobre 2024. Per ulteriori informazioni sui cluster `IPv6`, consulta the section called "IPv6". I cluster creati prima di ottobre 2024 utilizzano invece il seguente formato di endpoint.

## `Formato endpoint del cluster IPv4`

EKS crea un endpoint univoco nel seguente formato per ogni cluster selezionato `IPv4` nell'impostazione IP family (`ipFamily`) del cluster:

Example

AWS

Endpoint del cluster EKS public/private `eks-cluster.`*`region`*`.eks.amazonaws.com`

AWS GovCloud (US)

Endpoint del cluster public/private EKS `eks-cluster.region.eks.amazonaws.com`

Amazon Web Services in China

Endpoint del cluster public/private EKS `eks-cluster.region.amazonwebservices.com.cn`

> **Note**
>
> Prima di ottobre 2024, anche i cluster `IPv6` usavano questo formato di endpoint. Per questi cluster, sia l'endpoint pubblico che l'endpoint privato dispongono solo di indirizzi `IPv4` risolti da questo endpoint.

## Endpoint privato del cluster

Puoi abilitare l'accesso privato al server API Kubernetes in modo che tutte le comunicazioni tra i nodi di lavoro e il server API rimangano all'interno del VPC. È possibile limitare gli indirizzi IP che possono accedere al server API da Internet o disabilitare completamente l'accesso a Internet al server API.

> **Note**
>
> Poiché questo endpoint è destinato al server API Kubernetes e non è un AWS PrivateLink endpoint tradizionale per comunicare con un' AWS API, non viene visualizzato come endpoint nella console Amazon VPC.

Quando si abilita l'accesso privato all'endpoint per il cluster, Amazon EKS crea una zona ospitata privata Route 53 per conto dell'utente e la associa al VPC del cluster. Questa zona ospitata privata è gestita da Amazon EKS e non viene visualizzata nelle risorse Route 53 dell'account. Affinché la zona ospitata privata instradi correttamente il traffico verso il tuo server API, il VPC deve avere `enableDnsHostnames` e `enableDnsSupport` impostati su `true` e le opzioni DHCP impostate per il VPC devono includere `AmazonProvidedDNS` nell'elenco dei server dei nomi di dominio. Per ulteriori informazioni, consultare Visualizzazione e aggiornamento del supporto DNS per il VPC nella Guida per l'utente di Amazon VPC.

È possibile definire i requisiti di accesso all'endpoint del server API quando si crea un nuovo cluster e aggiornare l'accesso endpoint del server API per un cluster in qualsiasi momento.

> **Note**
>
> L'accesso all'endpoint controlla chi può raggiungere il server API Kubernetes. Se desideri anche gestire il modo in cui il traffico in uscita dal piano di controllo raggiunge le risorse del tuo VPC (come server webhook e provider OIDC), consulta. the section called "Uscita del piano di controllo"

## Modifica dell'accesso all'endpoint del cluster

Utilizza le procedure in questa sezione per modificare l'accesso all'endpoint per un cluster esistente. La tabella seguente mostra le combinazioni di accesso all'endpoint del server API supportate e il comportamento associato.

| Accesso pubblico all'endpoint | Accesso privato all'endpoint | Comportamento |
|---|---|---|
| Abilitato | Disabilitato | • Questo è il comportamento di default per nuovi cluster Amazon EKS.<br>• Le richieste API Kubernetes provenienti dal VPC del cluster (ad esempio, comunicazione tra nodo e piano di controllo) lasciano il VPC ma non la rete Amazon.<br>• Il server API del cluster è accessibile da Internet. Facoltativamente, puoi utilizzare l'elenco dei CIDR ad accesso pubblico per controllare l'accesso all'endpo int pubblico tramite un elenco di blocchi CIDR. Se limiti l'accesso a blocchi CIDR specifici, ti consigliamo di abilitare anche l'endpoint privato o di assicurarti che i blocchi CIDR specificati includano gli indirizzi da cui i nodi e i Fargate Pod (se li usi) accedono all'endpoint pubblico.<br>• Per il traffico tra i nodi di lavoro e il piano di controllo si applicano le tariffe standard per il trasferimento dei AWS dati. Amazon EKS assorbe i costi di trasferimento dei dati sul lato del piano di controllo della connessione. |

| Accesso pubblico all'endpoint | Accesso privato all'endpoint | Comportamento |
|---|---|---|
| | | Per ulteriori informazioni, consulta the section called "Prezzi di Amazon EKS". |
| Abilitato | Abilitato | • Le richieste API Kubernetes all'interno del VPC del cluster (ad esempio, comunicazione tra nodo di lavoro e piano di controllo) utilizzano l'endpoint VPC privato. Utilizzi le regole sul gruppo di sicurezza del cluster per controllare l'accesso all'endpoint privato.<br>• Il server API del cluster è accessibile da Internet. È possibile, facoltativamente, utilizzare l'elenco dei CIDR ad accesso pubblico per controllare l'accesso all'endpoint pubblico tramite un elenco di blocchi CIDR.<br>• Se utilizzi nodi ibridi con il tuo cluster Amazon EKS, non è consigliabile abilitare l'accesso agli endpoint sia pubblico che privato. Dato che i nodi ibridi sono in esecuzione all'esterno del VPC, risolveranno l'endpoin t del cluster negli indirizzi IP pubblici. Si consiglia di utilizzare l'accesso agli endpoint pubblici o privati per i cluster con nodi ibridi. |

| Accesso pubblico all'endpoint | Accesso privato all'endpoint | Comportamento |
| --- | --- | --- |
| Disabilitato | Abilitato | • Tutto il traffico verso il server API del cluster deve provenire dal VPC del cluster o da una rete connessa.<br>• Non esiste alcun accesso pubblico al server API da Internet. Tutti i comandi `kubectl` devono provenire dall'interno del VPC o da una rete connessa. Per le opzioni di connettività, consultare the section called "Accesso a un server API solo privato".<br>• Si utilizzano le regole sul gruppo di sicurezza del cluster per controllare l'accesso all'endpoint privato. Tieni presente che i CIDR di accesso pubblico non influiscono sull'endpoint privato.<br>• L'endpoint del server API del cluster viene risolto dai server DNS pubblici a un indirizzo IP privato dal VPC. In passato, l'endpoint poteva essere risolto solo dall'interno del VPC.<br><br>Se l'endpoint non viene risolto in un indirizzo IP privato all'interno del VPC per un cluster esistente, è possibile:<br>• Abilitare l'accesso pubblico e quindi disabilitarlo di nuovo. È necessario farlo solo una volta per un cluster e l'endpoint si risolverà in un indirizzo IP privato da quel punto in avanti.<br>• Aggiornare il cluster. |

Controlli di accesso agli endpoint

Tieni presente che ciascuno dei seguenti metodi per controllare l'accesso agli endpoint ha effetto solo sul rispettivo endpoint.

Gruppo di sicurezza del cluster

Il gruppo di sicurezza del cluster controlla due tipi di connessioni: connessioni all'API kubelet e all'endpoint privato. Le connessioni all'`kubelet`API vengono utilizzate nei comandi`kubectl attach`, `kubectl cp` `kubectl exec`, `kubectl logs`, e. `kubectl port-forward` Il gruppo di sicurezza del cluster non influisce sull'endpoint pubblico.

CIDR ad accesso pubblico

I CIDR ad accesso pubblico controllano l'accesso all'endpoint pubblico tramite un elenco di blocchi CIDR. Tieni presente che i CIDR ad accesso pubblico non influiscono sull'endpoint privato. I CIDR ad accesso pubblico si comportano in modo diverso sui cluster e sui `IPv6` `IPv4` cluster a seconda della data in cui sono stati creati, come descritto di seguito:

Blocchi CIDR nell'endpoint pubblico (cluster **`IPv6`**)

Puoi aggiungere blocchi `IPv6` e `IPv4` CIDR all'endpoint pubblico di un cluster `IPv6`, in quanto l'endpoint pubblico è dual-stack. Questo vale solo per i nuovi cluster con `ipFamily` impostato su `IPv6` creati a ottobre 2024 o successivamente. Puoi identificare questi cluster in base al nuovo nome di dominio dell'endpoint `api.aws`.

Blocchi CIDR nell'endpoint pubblico (cluster **`IPv4`**)

Puoi aggiungere blocchi CIDR `IPv4` all'endpoint pubblico di un cluster `IPv4`. Non puoi aggiungere blocchi CIDR `IPv6` all'endpoint pubblico di un cluster `IPv4`. Se provi a farlo, EKS restituirà il seguente messaggio di errore: `The following CIDRs are invalid in publicAccessCidrs`

Blocchi CIDR nell'endpoint pubblico (cluster **`IPv6`** creato prima di ottobre 2024)

Puoi aggiungere blocchi CIDR `IPv4` all'endpoint pubblico dei cluster `IPv6` precedenti creati prima di ottobre 2024. Puoi identificare questi cluster in base all'endpoint `eks.amazonaws.com`. Non puoi aggiungere blocchi CIDR `IPv6` all'endpoint pubblico dei cluster `IPv6` precedenti creati prima di ottobre 2024. Se provi a farlo, EKS restituirà il seguente messaggio di errore: `The following CIDRs are invalid in publicAccessCidrs`

## Accesso a un server API solo privato

Se è stato disabilitato l'accesso pubblico all'endpoint del server API Kubernetes del cluster, è possibile accedere al server di API solo dal VPC o da una rete connessa. Di seguito sono elencati alcuni possibili modi per accedere all'endpoint del server API Kubernetes:

Rete connessa

È possibile connettere la rete al VPC con un gateway di transito AWS o un'altra opzione di connettività e quindi utilizzare un computer nella rete connessa. Il gruppo di sicurezza del piano di controllo di Amazon EKS deve contenere le regole per consentire il traffico in ingresso sulla porta 443 dalla rete connessa.

Amazon EC2 Bastion Host

È possibile avviare un'istanza Amazon EC2 in una sottorete pubblica nel VPC del cluster e quindi accedere tramite SSH a tale istanza per eseguire comandi `kubectl`. Per ulteriori informazioni, consultare Bastion host Linux in AWS. Il gruppo di sicurezza del piano di controllo di Amazon EKS deve contenere le regole per consentire il traffico in ingresso sulla porta 443 dal bastion host. Per ulteriori informazioni, consulta the section called "Requisiti relativi al gruppo di sicurezza".

Quando `kubectl` configuri il tuo bastion host, assicurati di utilizzare AWS le credenziali già mappate alla configurazione RBAC del cluster o aggiungi il principio IAM che il tuo bastion utilizzerà alla configurazione RBAC prima di rimuovere l'accesso pubblico agli endpoint. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes" e the section called "Accesso non autorizzato o negato (kubectl) ".

AWS Cloud9 IDE

AWS Cloud9 è un ambiente di sviluppo integrato (IDE) basato su cloud che ti consente di scrivere, eseguire ed eseguire il debug del codice con un semplice browser. Puoi creare un IDE AWS Cloud9 nel VPC del tuo cluster e utilizzare l'IDE per comunicare con il tuo cluster. Per ulteriori informazioni, consulta Creazione di un ambiente in Cloud9. AWS È necessario assicurarsi che il gruppo di sicurezza del piano di controllo Amazon EKS contenga regole per consentire il traffico in ingresso sulla porta 443 dal gruppo di sicurezza IDE. Per ulteriori informazioni, consulta the section called "Requisiti relativi al gruppo di sicurezza".

Quando `kubectl` configuri l'IDE AWS Cloud9, assicurati di utilizzare AWS le credenziali già mappate alla configurazione RBAC del cluster o aggiungi l'entità IAM che il tuo IDE utilizzerà alla configurazione RBAC prima di rimuovere l'accesso pubblico agli endpoint. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes" e the section called "Accesso non autorizzato o negato (kubectl) ".

AWS CloudShell

Scegli Connect nella pagina dei dettagli del cluster nella console Amazon EKS. Per i cluster privati, CloudShell avvia un ambiente VPC in grado di raggiungere l'endpoint del server API

privato del cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".

# Modifica questa pagina su GitHub

## Configurare l'accesso di rete all'endpoint del server API del cluster

È possibile modificare l'accesso agli endpoint del server API del cluster utilizzando Console di gestione AWS o la AWS CLI nelle sezioni seguenti.

### Configura l'accesso agli endpoint - AWS console

1. Aprire la Console Amazon EKS.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegli la scheda Networking e scegli Gestisci accesso all'endpoint.
4. Per Accesso privato, scegli se attivare o disattivare l'acceso privato per l'endpoint del server API Kubernetes del cluster. Se abiliti l'accesso privato, le richieste API Kubernetes che provengono dal VPC del cluster utilizzano l'endpoint VPC privato. Per disabilitare l'accesso pubblico è necessario abilitare l'accesso privato.
5. Per Accesso pubblico, scegli se attivare o disattivare l'acceso pubblico per l'endpoint del server API Kubernetes del cluster. Se disabiliti l'accesso pubblico, il server API Kubernetes del cluster può ricevere solo richieste dal VPC del cluster.
6. (Facoltativo) Se abilitati l'Accesso pubblico, puoi specificare quali indirizzi Internet possono comunicare con l'endpoint pubblico. Seleziona Impostazioni avanzate. Immettere un blocco CIDR, ad esempio *`203.0.113.5/32`*. L'intervallo non può includere indirizzi riservati. Puoi immettere intervalli aggiuntivi selezionando Aggiungi origine. Puoi specificare un numero massimo di intervalli CIDR. Per ulteriori informazioni, consulta the section called "Service Quotas". Se non si specificano gli intervalli, l'endpoint del server API pubblico riceve richieste da tutti gli indirizzi IP sia per `IPv4` (`0.0.0.0/0`) che in aggiunta `IPv6` (`::/0`) per il cluster `IPv6` a doppio stack. Se limitati l'accesso all'endpoint pubblico utilizzando gli intervalli CIDR, ti consigliamo di abilitare anche l'accesso agli endpoint privati in modo che i nodi e Fargate Pods (se utilizzati) possano comunicare con il cluster. Senza l'endpoint privato abilitato, le origini CIDR dell'endpoint di accesso pubblico devono includere le origini di uscita dal VPC. Ad esempio, se si dispone di un nodo in una sottorete privata che comunica su Internet tramite un gateway NAT, sarà necessario aggiungere l'indirizzo IP in uscita del gateway NAT come parte di un blocco CIDR nella whitelist nell'endpoint pubblico.

7. Scegliere Update (Aggiorna) per terminare.

## Configura l'accesso agli endpoint - AWS CLI

Completa i seguenti passaggi utilizzando la versione AWS CLI `1.27.160` o successiva. È possibile verificare la versione corrente con `aws --version`. Per installare o aggiornare la AWS CLI, vedi Installazione della AWS CLI.

1. Aggiorna l'accesso agli endpoint del server API del cluster con il seguente comando AWS CLI. Sostituire il nome del cluster e i valori di accesso dell'endpoint desiderati. Se lo impostiendpointPublicAccess=true, puoi (facoltativamente) inserire un singolo blocco CIDR o un elenco separato da virgole di blocchi CIDR per. publicAccessCidrs I blocchi non possono includere indirizzi riservati. Se si specificano blocchi CIDR, l'endpoint del server API pubblico riceverà solo le richieste dai blocchi elencati. Puoi specificare un numero massimo di intervalli CIDR. Per ulteriori informazioni, consulta the section called "Service Quotas". Se limiti l'accesso all'endpoint pubblico utilizzando i blocchi CIDR, è consigliabile abilitare anche l'accesso agli endpoint privati in modo che i nodi e i Pods Fargate (se utilizzati) possano comunicare con il cluster. Senza l'endpoint privato abilitato, le origini CIDR dell'endpoint di accesso pubblico devono includere le origini di uscita dal VPC. Ad esempio, se si dispone di un nodo in una sottorete privata che comunica su Internet tramite un gateway NAT, sarà necessario aggiungere l'indirizzo IP in uscita del gateway NAT come parte di un blocco CIDR nella whitelist nell'endpoint pubblico. Se non specificate alcun blocco CIDR, l'endpoint del server API pubblico riceve le richieste da tutti (0.0.0). 0/0) Indirizzi IP e in aggiunta `IPv6` (`::/0`) per cluster dual-stack`IPv6`.

> **Note**
>
> Il comando seguente consente l'accesso privato e l'accesso pubblico da un singolo indirizzo IP per l'endpoint del server API. Sostituire *203.0.113.5/32* con un singolo blocco CIDR o un elenco separato da virgole di blocchi CIDR a cui si desidera limitare l'accesso alla rete.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

Di seguito viene riportato un output di esempio:

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

2. Monitorare lo stato di aggiornamento dell'accesso all'endpoint con il comando seguente, utilizzando il nome del cluster, e aggiornare l'ID restituito dal comando precedente. L'aggiornamento è completo quando lo stato è illustrato come `Successful`.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

Di seguito viene riportato un output di esempio.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
```

```
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

# Modifica questa pagina su GitHub

# Implementazione dei nodi Windows su cluster EKS

Informazioni su come abilitare e gestire il supporto Windows per i cluster Amazon EKS per l'esecuzione di container Windows insieme a container Linux.

## Considerazioni

Prima di implementare i nodi di Windows, tenere conto delle seguenti considerazioni.

- La modalità automatica EKS non supporta i nodi Windows
- Puoi utilizzare la rete host sui nodi Windows utilizzando i Pod `HostProcess` . Per ulteriori informazioni, consulta Creare un Windows HostProcess Pod nella documentazione di Kubernetes.
- I cluster Amazon EKS devono contenere uno o più nodi Linux o Fargate per eseguire i pod del sistema core che funzionano solo su Linux, come CoreDNS.
- I log degli eventi `kubelet` e `kube-proxy` sono reindirizzati al log degli eventi `EKS Windows` e sono impostati su un limite di 200 MB.

- Non è possibile utilizzare Assegna gruppi di sicurezza a singoli pod con i pod in esecuzione su nodi Windows.
- Non è possibile utilizzare reti personalizzate con nodi Windows.
- Non è possibile utilizzare `IPv6` con nodi Windows.
- I nodi di Windows supportano un'interfaccia di rete elastica per nodo. Per impostazione predefinita, il numero di pod che è possibile eseguire per ogni nodo Windows è uguale al numero di indirizzi IP (meno uno) disponibili per interfaccia di rete elastica relativa al tipo di istanza del nodo. Per ulteriori informazioni, consulta la sezione relativa agli indirizzi IP per interfaccia di rete per tipo di istanza nella Guida per l'utente di Amazon EC2.
- In un cluster Amazon EKS, un singolo servizio con un bilanciatore del carico può supportare fino a 1024 pod back-end. Ogni pod ha il proprio indirizzo IP univoco. Il limite precedente di 64 pod non è più valido, dopo un aggiornamento di Windows Server a partire da Build del sistema operativo 17763.2746.
- I container Windows non sono supportati per i pod Amazon EKS su Fargate.
- Non è possibile utilizzare Amazon EKS Hybrid Nodes con Windows come sistema operativo per l'host.
- Non è possibile recuperare i log dal pod `vpc-resource-controller`. In precedenza era possibile quando si implementava il controller sul piano dati.
- Esiste un periodo di raffreddamento prima che un indirizzo `IPv4` venga assegnato a un nuovo pod. In questo modo, si impedisce che il traffico vada verso un pod precedente con lo stesso indirizzo `IPv4` a causa della mancata validità delle regole `kube-proxy`.
- L'origine del controller è gestita su. GitHub Per contribuire o segnalare problemi al controller, visita il progetto su GitHub.
- Quando specifichi un ID AMI personalizzato per i gruppi di nodi gestiti da Windows, aggiungilo `eks:kube-proxy-windows` alla mappa di configurazione di AWS IAM Authenticator. Per ulteriori informazioni, consulta the section called “Limiti e condizioni quando si specifica un ID AMI”.
- Se la conservazione degli indirizzi IPv4 disponibili è fondamentale per la sottorete, consulta la EKS Best Practices Guide - Windows Networking IP Address Management per maggiori indicazioni.
- Considerazioni relative a voci di accesso EKS
  - Le voci di accesso da utilizzare con i nodi Windows richiedono il tipo di `EC2_WINDOWS`. Per ulteriori informazioni, consulta the section called “Creare voci di accesso”.

    Per creare una voce di accesso per un nodo Windows:

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/<role-name> --type EC2_Windows
```

## Prerequisiti

- Un cluster esistente.
- Il tuo cluster deve avere almeno un nodo Linux (ne consigliamo almeno due) o un pod Fargate per eseguire CoreDNS. Se si abilita il supporto Windows legacy, è necessario utilizzare un nodo Linux (non è possibile utilizzare un pod Fargate) per eseguire CoreDNS.
- Un ruolo IAM del cluster Amazon EKS esistente.

## Abilitazione del supporto Windows

1. Se non disponi di nodi Amazon Linux nel cluster e utilizzi gruppi di sicurezza per i pod, vai al passaggio successivo. Altrimenti, conferma che la policy gestita di `AmazonEKSVPCResourceController` è collegata al tuo ruolo del cluster. Sostituisci *`eksClusterRole`* con il nome del ruolo del cluster.

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   Di seguito viene riportato un output di esempio:

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSClusterPolicy"
           },
           {
               "PolicyName": "AmazonEKSVPCResourceController",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSVPCResourceController"
           }
       ]
   }
   ```

   Se la policy è collegata, come nell'output precedente, salta il passaggio successivo.

2. Collega la policy AmazonEKSVPCResourceControllergestita al tuo ruolo IAM del cluster Amazon EKS. Sostituisci *eksClusterRole* con il nome del ruolo del cluster.

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSVPCResourceController
```

3. Aggiorna il VPC CNI ConfigMap per abilitare Windows IPAM. Nota, se il VPC CNI è installato sul tuo cluster utilizzando un grafico Helm o come Amazon EKS, Add-on potresti non essere in grado di modificare direttamente il. ConfigMap Per informazioni sulla configurazione di Amazon EKS Add-ons, consultathe section called "Campi che puoi personalizzare".

   a. Crea un file denominato *vpc-resource-controller-configmap.yaml* con i seguenti contenuti.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

   b. Applica la `ConfigMap` al cluster.

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

4. Se il cluster ha la modalità di autenticazione impostata per abilitare configmap `aws-auth`:

   - Verifica che `aws-auth` `ConfigMap` contenga una mappatura per il ruolo di istanza del nodo Windows per includere il gruppo di autorizzazioni `eks:kube-proxy-windows` RBAC. Puoi eseguire la verifica eseguendo il comando seguente.

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

     Di seguito viene riportato un output di esempio.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
```

```
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws: iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

Dovresti vedere `eks:kube-proxy-windows` elencato nei gruppi. Se il gruppo non è specificato, devi aggiornare `ConfigMap` o crearla per includere il gruppo richiesto. Per ulteriori informazioni su `aws-auth` `ConfigMap`, consulta the section called "Applica aws-auth al tuo cluster `ConfigMap`".

5. Se il cluster ha la modalità di autenticazione impostata per disabilitare configmap `aws-auth`, è possibile utilizzare le voci di accesso EKS. La creazione di un nuovo ruolo del nodo da utilizzare con le istanze Windows ed EKS creerà automaticamente una voce di accesso di tipo `EC2_WINDOWS`.

## Implementazione di pods Windows

Quando si implementano i pod nel cluster, è necessario specificare il sistema operativo utilizzato se si esegue una combinazione di tipi di nodi.

Per i pod Linux, utilizzare il seguente testo del selettore dei nodi nei manifesti.

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Per i pod Windows, utilizzare il seguente testo del selettore dei nodi nei manifesti.

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

È possibile implementare un'applicazione di esempio per vedere i selettori dei nodi in uso.

## Supporto di maggiore densità di pod sui nodi di Windows

Su Amazon EKS a ciascun pod è assegnato un indirizzo `IPv4` dal tuo VPC. Per questo motivo, il numero di pod che è possibile distribuire su un nodo è vincolato agli indirizzi IP disponibili, anche se ci sono risorse sufficienti per eseguire più pod sul nodo. Poiché un nodo di Windows supporta una sola interfaccia di rete elastica, per impostazione predefinita, il numero massimo di indirizzi IP disponibili su un nodo di Windows è pari a:

```
Number of private IPv4 addresses for each interface on the node - 1
```

Un indirizzo IP è utilizzato come indirizzo IP principale dell'interfaccia di rete, quindi non può essere assegnato ai pod.

È possibile abilitare una maggiore densità di pod sui nodi Windows abilitando la delega del prefisso IP. Questa funzionalità consente di assegnare un prefisso`/28` `IPv4` all'interfaccia di rete principale, anziché assegnare indirizzi `IPv4` secondari. L'assegnazione di un prefisso IP aumenta il numero massimo di indirizzi `IPv4` disponibili sul nodo:

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

Con questo numero di indirizzi IP disponibili di gran lunga maggiore, gli indirizzi IP disponibili non dovrebbero limitare la capacità di scalare il numero di pod sui tuoi nodi. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP".

# Disabilitare il supporto Windows

1. Se il cluster contiene nodi Amazon Linux e si utilizzano gruppi di sicurezza per i pod, saltare questo passaggio.

   Rimuovi la policy IAM gestita `AmazonVPCResourceController` dal ruolo del cluster. Sostituire *`eksClusterRole`* con il nome del ruolo del cluster.

   ```
   aws iam detach-role-policy \
       --role-name eksClusterRole \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
   ```

2. Disabilita Windows IPAM nella ConfigMap per `amazon-vpc-cni`.

```
kubectl patch configmap/amazon-vpc-cni \
                    -n kube-system \
                    --type merge \
                    -p '{"data":{"enable-windows-ipam":"false"}}'
```

# Implementazione di cluster privati con accesso limitato a Internet

Questo argomento descrive come distribuire un cluster Amazon EKS distribuito sul AWS cloud, ma non dotato di accesso a Internet in uscita. Se hai un cluster locale su AWS Outposts, consultathe section called "Nodi", invece di questo argomento.

> **Note**
>
> Questo argomento riguarda l'uscita dal piano di dati (nodi e carichi di lavoro senza accesso a Internet in uscita). Per controllare in che modo il traffico in uscita dal piano di controllo del cluster raggiunge il tuo VPC, consulta. the section called "Uscita del piano di controllo"

Se non hai familiarità con la rete Amazon EKS, consulta la rete in De-mystifying cluster per i nodi di lavoro di Amazon EKS. Se il cluster non dispone di accesso a Internet in uscita, deve soddisfare i seguenti requisiti:

## Requisiti dell'architettura del cluster

- Il tuo cluster deve estrarre immagini da un registro di container che si trova nel tuo VPC. Puoi creare un Amazon Elastic Container Registry nel VPC e copiare al suo interno le immagini dei container per i nodi da cui estrarre. Per ulteriori informazioni, consulta the section called "Copia di un'immagine in un repository".
- Il tuo cluster deve avere l'accesso privato all'endpoint abilitato. Ciò è obbligatorio per la registrazione dei nodi con l'endpoint del cluster. L'accesso pubblico all'endpoint è facoltativo. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

# Requisiti dei nodi

- Self-managed I nodi Linux e Windows devono includere i seguenti argomenti di bootstrap prima di essere avviati. Questi argomenti ignorano l'introspezione Amazon EKS e non richiedono l'accesso all'API Amazon EKS dal VPC.

  a. Determina l'endpoint del cluster con il comando seguente. Sostituisci *my-cluster* con il nome del cluster.

  ```
  aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
  ```

  Di seguito viene riportato un output di esempio.

  ```
  https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
  ```

  b. Determina il valore dell'autorità di certificazione del cluster con il seguente comando. Sostituisci *my-cluster* con il nome del cluster.

  ```
  aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --
  output text
  ```

  L'output è una stringa molto lunga.

  c. Sostituisci i valori di `apiServerEndpoint` e `certificateAuthority` nell' NodeConfig oggetto con i valori restituiti nell'output dei comandi precedenti. Per ulteriori informazioni su come specificare gli argomenti di bootstrap all'avvio di nodi Amazon Linux 2023 autogestiti, consulta e. the section called "Amazon Linux" the section called "Windows"

  - Per i nodi Linux:

  ```
  ---
  MIME-Version: 1.0
  Content-Type: multipart/mixed; boundary="BOUNDARY"

  --BOUNDARY
  Content-Type: application/node.eks.aws

  ---
  apiVersion: node.eks.aws/v1alpha1
  kind: NodeConfig
  spec:
    cluster:
  ```

```
    name: my-cluster
    apiServerEndpoint: https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-
code.eks.amazonaws.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    ...
```

Per argomenti aggiuntivi, consulta lo script bootstrap su. https://github.com/awslabs/amazon-eks-ami/blob/main/templates/al2/runtime/bootstrap.sh GitHub

- Per i nodi Windows:

> **Note**
>
> Se utilizzi il servizio personalizzato CIDR, devi specificarlo utilizzando il parametro `-ServiceCIDR`. In caso contrario, la risoluzione DNS per i pod nel cluster avrà esito negativo.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

Per ulteriori argomenti, consulta.the section called "Parametri di configurazione dello script di bootstrap".

- `aws-auth ConfigMap` del tuo cluster deve essere creata all'interno del tuo VPC. Per ulteriori informazioni sulla creazione e l'aggiunta di voci a `aws-auth ConfigMap`, digita `eksctl create iamidentitymapping --help` nel terminale. Se `ConfigMap` non esiste sul server, `eksctl` la creerà quando usi il comando per aggiungere una mappatura dell'identità.

## Requisiti del pod

- Pod Identity: i pod configurati con EKS Pod Identity acquisiscono le credenziali dall'API EKS Auth. Se non è presente un accesso a Internet in uscita, è necessario creare e utilizzare un endpoint VPC per l'API EKS Auth: `com.amazonaws.region-code.eks-auth`. Per ulteriori informazioni sugli endpoint VCP EKS ed EKS Auth, consulta the section called " AWS PrivateLink".
- IRSA - I pod configurati con ruoli IAM per gli account di servizio acquisiscono le credenziali da una chiamata API AWS Security Token Service (AWS STS). Se non c'è accesso a Internet in uscita, devi creare e utilizzare un endpoint AWS STS VPC nel tuo VPC. La maggior parte degli AWS `v1` SDK utilizza l'endpoint AWS STS globale per impostazione predefinita

(`sts.amazonaws.com`), che non utilizza l'endpoint STS VPC. AWS Per utilizzare l'endpoint AWS STS VPC, potrebbe essere necessario configurare l'SDK per utilizzare l'endpoint STS regionale (). AWS `sts.region-code.amazonaws.com` Per ulteriori informazioni, consulta the section called "Endpoint STS".

Per configurare IRSA, devi anche raggiungere l'endpoint OIDC del cluster dall'interno del VPC discovery/JWKS . Ad esempio, lo raggiungi quando crei il provider di identità IAM OIDC del cluster o esegui i tuoi validatori di token. Se non c'è accesso a Internet in uscita, crea e utilizza un endpoint VPC per l'endpoint OIDC del cluster:. `com.amazonaws.region-code.oidc-eks` Questo è separato dall'endpoint STS VPC. AWS STS recupera i JWKS del cluster dall'interno AWS, quindi l'endpoint OIDC privatizza il traffico OIDC, mentre l'endpoint VPC-originated STS privatizza la chiamata. `AssumeRoleWithWebIdentity` Per ulteriori informazioni, consulta the section called "Accedi all'endpoint OIDC del cluster utilizzando AWS PrivateLink".

- Le sottoreti VPC del cluster devono disporre di un endpoint di interfaccia VPC per tutti i servizi a cui i Pod devono accedere. AWS Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un endpoint VPC di interfaccia. Nella tabella seguente sono elencati alcuni servizi ed endpoint di uso comune. Per un elenco completo degli endpoint, consulta Servizi AWS che si integrano con AWS PrivateLink nella Guida di AWS PrivateLink .

  Ti consigliamo di abilitare i nomi DNS privati per i tuoi endpoint VPC, in questo modo i carichi di lavoro possono continuare a utilizzare gli endpoint di servizio pubblico senza problemi. AWS

| Servizio | Endpoint |
| --- | --- |
| Amazon EC2 | com.amazonaws. *region-code* .ec2 |
| Amazon Elastic Container Registry (per estrarre immagini di container) | com.amazonaws. *region-code* ecr.api, com.amazonaws. *region-code* .ecr.dkr e com.amazonaws. *region-code* .s3 |
| Application Load Balancer e Network Load Balancer di Amazon | com.amazonaws. *region-code* . bilanciam ento del carico elastico |
| (Facoltativo) (obbligatorio per il tracciamento inviato a AWS X-Ray ) AWS X-Ray | com.amazonaws. *region-code* .radiografia |

| Servizio | Endpoint |
| --- | --- |
| (Facoltativo) Amazon SSM (necessario per l'agente SSM per le attività di gestione dei nodi. Alternativa a SSH) | com.amazonaws. *region-code* .ssm |
| Amazon CloudWatch Logs (richiesto per i log di nodi e pod inviati ad Amazon CloudWatch Logs) | com.amazonaws. *region-code* .registri |
| AWS Security Token Service (richiesto quando si utilizzano i ruoli IAM per gli account di servizio) | com.amazonaws. *region-code* .st |
| Endpoint OIDC del cluster Amazon EKS (necessario per configurare i ruoli IAM per gli account di servizio dall'interno del VPC) | com.amazonaws. *region-code* .oidc-eks |
| Amazon EKS Auth (necessario quando si utilizzano le associazioni Pod Identity) | com.amazonaws. *region-code* .eks-auth |
| Amazon EKS | con amazonaws. *region-code* .eks |
| Amazon Route 53 | com.amazonaws.route53 |

- Qualsiasi nodo autogestito deve essere implementato in sottoreti con gli endpoint di interfaccia VPC richiesti. Se crei un gruppo di nodi gestiti, il gruppo di sicurezza dell'endpoint di interfaccia VPC deve consentire il CIDR per le sottoreti oppure dovrai aggiungere il gruppo di sicurezza del nodo creato al gruppo di sicurezza dell'endpoint di interfaccia VPC.
- Storage EFS: se i tuoi pod utilizzano volumi Amazon EFS, prima di distribuire l'archivio di un file system elastico con Amazon EFS, il file kustomization.yaml del driver deve essere modificato per impostare le immagini del contenitore in modo che utilizzino la stessa regione del cluster Amazon EKS. AWS
- Se utilizzi l'AMI ottimizzata EKS, devi abilitare l'endpoint `ec2` nella tabella precedente. In alternativa, esiste la possibilità di impostare manualmente il nome DNS del nodo. L'AMI ottimizzata utilizza le API EC2 per impostare automaticamente il nome DNS del nodo.
- Puoi utilizzare il AWS Load Balancer Controller per distribuire AWS Application Load Balancer (ALB) e Network Load Balancer nel tuo cluster privato. Quando lo implementi, devi usare i flag

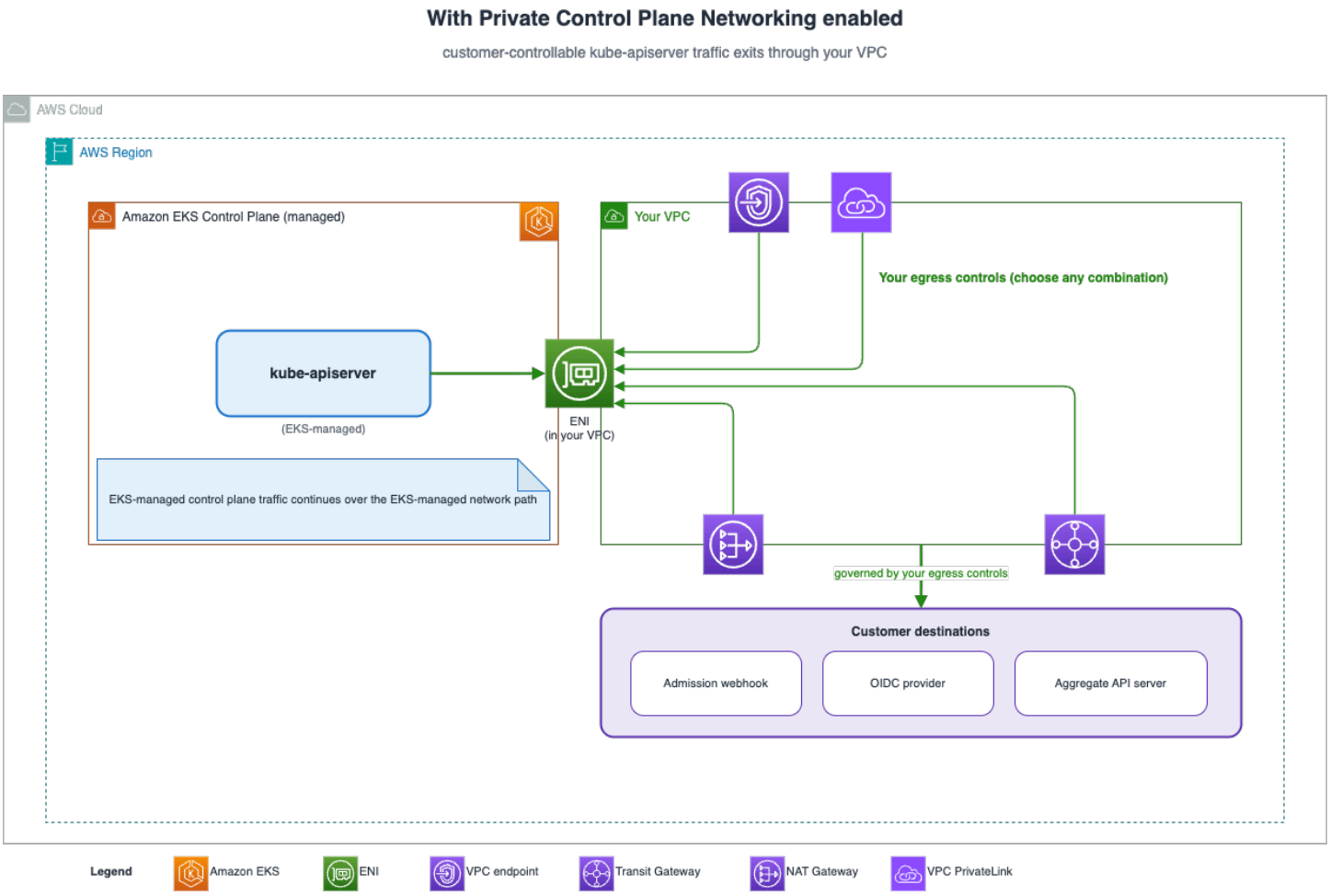
della riga di comando per impostare`enable-shield`, `enable-waf` e `enable-wafv2` su fasle. Il rilevamento dei certificati con i nomi host degli oggetti in entrata non è supportato. Questo perché il controller deve raggiungere AWS Certificate Manager, che non dispone di un endpoint di interfaccia VPC.

Il controller supporta bilanciatori di carico di rete destinazioni IP, necessari per l'utilizzo con Fargate. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di applicazione" e the section called "Creazione di un Network Load Balancer".

- Cluster Autoscaler è supportato. Quando si implementano i pod di Cluster Autoscaler, assicurarsi che la riga di comando includa `--aws-use-static-instance-list=true`. Per ulteriori informazioni, consulta Use Static Instance List on. GitHub Il nodo di lavoro VPC deve includere anche l'endpoint AWS STS VPC e l'endpoint VPC con scalabilità automatica.
- Alcuni prodotti software per container utilizzano chiamate API che accedono al Marketplace Metering Service per monitorare l'utilizzo. AWS I cluster privati non consentono queste chiamate, pertanto questi tipi di container non possono essere utilizzati nei cluster privati.

## Configurazione del routing in uscita dal piano di controllo

Per impostazione predefinita, Amazon EKS gestisce la rete in uscita dal piano di controllo di Kubernetes alle risorse nel tuo VPC. Usa  il routing in uscita del piano di controllo  per modificare questo comportamento e gestire tu stesso il percorso di rete. Questo ti dà il pieno controllo sul modo in cui il traffico proveniente dalle interfacce di rete elastiche (ENI) del piano di controllo raggiunge le tue risorse VPC. È possibile eseguire il routing attraverso i propri gateway NAT, firewall o dispositivi di ispezione.




## Modalità di routing in uscita

Amazon EKS supporta le seguenti modalità di routing in uscita dal piano di controllo:

| Modalità | Description |
|---|---|
| `AWS_MANAGED` | Comportamento predefinito. Amazon EKS gestisce il percorso di uscita dagli ENI del piano di controllo. Non è necessario configurare gateway NAT o altra infrastruttura di routing per controllare il traffico aereo. |
| `CUSTOMER_ROUTED` | Gestisci il percorso di uscita dal piano di controllo nelle tue sottoreti VPC. Sei responsabile di garantire che il piano di controllo possa raggiunge re gli endpoint richiesti (come server webhook, provider OIDC e altre risorse). Fornisci un percorso di uscita, ad esempio un gateway NAT, un'istanza NAT, un gateway di transito o un'appliance firewall. È inoltre |

| Modalità | Description |
| --- | --- |
| | possibile configurare la tabella di routing, l'ACL di rete e le regole del gruppo di sicurezza che consentono questo traffico. |

> **Important**
>
> In `CUSTOMER_ROUTED` modalità, sei responsabile di garantire la corretta connettività di rete dal piano di controllo. Le configurazioni errate nella rete VPC possono causare il fallimento delle operazioni del piano di controllo. Queste configurazioni errate includono un percorso di uscita mancante, ACL di rete restrittivi o gruppi di sicurezza errati. Le operazioni interessate includono le chiamate webhook di ammissione e l'autenticazione OIDC.

## Prerequisiti

Il tuo VPC e le tue sottoreti devono soddisfare i requisiti di rete standard di Amazon EKS. Per ulteriori informazioni, consulta the section called “Requisiti del VPC e delle sottoreti”.

In `CUSTOMER_ROUTED` modalità, il server API Kubernetes invia il traffico rivolto ai clienti in uscita attraverso le interfacce di rete tra account. Amazon EKS crea già queste interfacce nelle tue sottoreti per la comunicazione tra piano di controllo e nodo. Questo traffico include le chiamate ai webhook di ammissione e ai provider OIDC. Amazon EKS non crea interfacce di rete di uscita separate. Questa modalità modifica il modo in cui vengono utilizzate le interfacce esistenti. Le sottoreti che contengono queste interfacce devono soddisfare i seguenti requisiti:

- Le sottoreti devono avere un percorso verso gli endpoint che il piano di controllo deve raggiungere (come server webhook e provider OIDC). Per gli endpoint esterni al tuo VPC, di solito si tratta di un percorso predefinito verso un dispositivo di uscita. Il percorso predefinito è `0.0.0.0/0` per IPv4 e per IPv6. `::/0` Il dispositivo di uscita può essere un gateway NAT, un'istanza NAT, un firewall o un gateway di transito verso un VPC di uscita centralizzato. La scelta del dispositivo di uscita è tua; Amazon EKS richiede solo che il percorso funzioni.
- I gruppi di sicurezza sulle interfacce di rete con più account devono consentire il traffico in uscita sulle porte richieste dai carichi di lavoro (ad esempio, la porta 443 per webhook e provider OIDC).
- Gli ACL di rete sulle sottoreti devono consentire il traffico in uscita e il corrispondente intervallo di porte temporanee in entrata per il traffico di ritorno.

In `CUSTOMER_ROUTED` modalità, il piano di controllo risolve i nomi host utilizzando la configurazione DNS del VPC. Ciò consente al piano di controllo di raggiungere gli endpoint nelle zone private ospitate di Route 53 e il DNS locale inoltrato tramite gli endpoint Route 53 Resolver.

- Il set di opzioni DHCP VPC deve essere incluso nell'elenco dei server dei nomi di dominio. `AmazonProvidedDNS` Ciò è necessario affinché il piano di controllo risolva i nomi DNS all'interno del VPC. Se il cluster utilizza endpoint webhook esterni o provider OIDC con nomi DNS pubblici, il resolver deve risolvere anche i nomi host pubblici. Assicurati che il resolver sia in grado di gestire sia la risoluzione VPC che quella DNS pubblica.

La tabella seguente riassume il traffico che il piano di controllo invia attraverso il tuo VPC in modalità: `CUSTOMER_ROUTED`

| Traffico | Destinazione | Porta | Note |
|---|---|---|---|
| Webhook di ammissione | Endpoint Webhook (URL definiti dal cliente) | 443 (in genere) | Solo se i webhook sono configurati. Lascia il VPC attraverso il dispositivo di uscita se l'endpoint è esterno. |
| Scoperta dell'OIDC | URL dell'emittente OIDC | 443 | Solo se è configurato un provider OIDC. Lascia il VPC attraverso il dispositivo di uscita se l'emittente è esterno. |
| Server API aggregati | Endpoint del server API del cliente | 443 | Solo se configurato. Lascia il VPC attraverso il dispositivo di uscita se l'endpoint è esterno. |
| API Kubelet | Indirizzi IP del nodo di lavoro | 10250 | Si tratta del traffico tra il piano di controllo e i nodi attraverso il cluster ENI; non attraversa il dispositivo di uscita. Richiede che le tabelle di routing, |

| Traffico | Destinazione | Porta | Note |
| --- | --- | --- | --- |
| | | | i gruppi di sicurezza e gli ACL di rete consentano il traffico attraverso il cluster ENI tra il piano di controllo e i nodi. |

> **Note**
>
> Solo il traffico elencato in questa tabella è influenzato dalla configurazione dell'uscita. EKS-managed il traffico del piano di controllo (ad esempio la comunicazione con etcd, CloudWatch Logs e servizi EKS interni) continua attraverso il percorso di rete AWS gestito e non è influenzato dalla configurazione del VPC.

## Crea un cluster con i dati di uscita indirizzati dal cliente

È possibile specificare la modalità di uscita del piano di controllo quando si crea un nuovo cluster.

Example

AWS CLI

Eseguire il seguente comando seguente. Sostituisci i *placeholder values* con i valori in tuo possesso.

```
aws eks create-cluster \
    --name my-cluster \
    --role-arn arn:aws:iam::111122223333:role/myAmazonEKSClusterRole \
    --resources-vpc-config "subnetIds=subnet-ExampleID1,subnet-
ExampleID2,securityGroupIds=sg-ExampleID1,controlPlaneEgressMode=CUSTOMER_ROUTED" \
    --kubernetes-network-config "ipFamily=ipv4" \
    --region region-code
```

È possibile utilizzarlo `ipFamily=ipv6` per i cluster IPv6. Quando utilizzi IPv6 con `CUSTOMER_ROUTED` modalità, assicurati che le sottoreti dispongano di un gateway Internet di sola uscita per il traffico IPv6 oltre a un gateway NAT per il traffico IPv4.

Example

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Add cluster (Aggiungi cluster), quindi scegli Create (Crea).
3. Nella pagina Rete, per Control plane egress, seleziona  Customer routed.
4. Completa la configurazione del cluster rimanente e scegli  Crea.

Per AWS CloudFormation, imposta `ControlPlaneEgressMode: CUSTOMER_ROUTEDResourcesVpcConfig`. Il supporto Terraform per questo campo sarà disponibile in una versione futura del AWS Provider.

> **Note**
>
> Il passaggio a `CUSTOMER_ROUTED` è un'operazione unidirezionale. Dopo aver abilitato l'uscita indirizzata dal cliente su un cluster, non è possibile tornare a. `AWS_MANAGED`

## Aggiornare un cluster esistente

È possibile modificare la modalità di uscita del piano di controllo su un cluster esistente utilizzando il comando. `update-cluster-config`

```
aws eks update-cluster-config \
    --name my-cluster \
    --resources-vpc-config "controlPlaneEgressMode=CUSTOMER_ROUTED" \
    --region region-code
```

Monitora lo stato dell'aggiornamento:

```
aws eks describe-update \
    --name my-cluster \
    --update-id update-id \
    --region region-code
```

L'aggiornamento è completo quando viene visualizzato lo stato`Successful`. Il tipo di aggiornamento è`ControlPlaneEgressUpdate`. L'aggiornamento viene in genere completato entro 10 minuti.

> **Important**
>
> Il passaggio a `CUSTOMER_ROUTED` è un'operazione unidirezionale. Dopo aver abilitato l'uscita indirizzata dal cliente su un cluster, non è possibile tornare a. `AWS_MANAGED`
> Prima di passare, verifica che il tuo VPC soddisfi i requisiti in. the section called "Prerequisiti"
> Se il piano di controllo perde la connettività agli endpoint richiesti dopo l'aggiornamento, operazioni come le chiamate webhook di ammissione e l'autenticazione OIDC possono fallire.

## Considerazioni sull'IPv6

Se si esegue un cluster IPv6 con uscita indirizzata dal cliente, è necessario configurare i percorsi di uscita IPv4 e IPv6.

Quando si esegue `ipFamily=ipv6` un cluster `CUSTOMER_ROUTED` IPv6 () con uscita:

- Agli ENI del piano di controllo vengono assegnati indirizzi IPv4 e IPv6.
- È necessario configurare i percorsi di uscita IPv4 e IPv6:
  - IPv4: una route predefinita (`0.0.0.0/0`) verso il dispositivo di uscita (ad esempio, un gateway NAT).
  - IPv6: un `::/0` percorso verso un dispositivo di uscita IPv6 (ad esempio, un gateway Internet solo in uscita).
- I gruppi di sicurezza e i NACL devono consentire il traffico su entrambe le versioni IP.
- Se il tuo provider OIDC o gli endpoint webhook lo sono IPv4-only, assicurati che il NAT IPv4 sia funzionante.

## Considerazioni

Tieni presente i seguenti punti quando utilizzi l'uscita dal piano di controllo indirizzato dal cliente:

- La tua responsabilità: in `CUSTOMER_ROUTED` modalità, sei il proprietario del percorso di rete dal piano di controllo agli endpoint esterni. Se il percorso si interrompe, le operazioni del piano di controllo che dipendono da esso (come le chiamate webhook di ammissione e l'autenticazione OIDC) possono fallire finché non si ripristina la connettività.

- VPC-internal il traffico non viene alterato: il traffico tra il piano di controllo e i nodi (ad esempio, l'API kubelet sulla porta 10250) attraverso il cluster ENI non dipende dal dispositivo di uscita utilizzato.
- EKS Auto Mode: il routing in uscita del piano di controllo funziona allo stesso modo sui cluster Standard e Auto Mode perché l'architettura del piano di controllo è identica.
- Funzionalità EKS: le funzionalità EKS (come ArgoCD, ACK e KRO) vengono eseguite in un'infrastruttura gestita separata. AWS Il traffico proveniente dai controller EKS Capabilities non viene indirizzato attraverso il tuo VPC tramite questa funzionalità.
- Osservabilità: Se abiliti i log di flusso VPC sulle sottoreti del tuo VPC o del cluster, puoi osservare il traffico in uscita che attraversa il tuo VPC. Ciò include le chiamate agli endpoint webhook e OIDC. Se i log di flusso VPC non sono abilitati, questo traffico non viene registrato.

## Chiave di condizione IAM

Amazon EKS supporta la chiave `eks:controlPlaneEgressMode` condizionale. Puoi utilizzare questa chiave nelle policy IAM o nelle policy di controllo dei servizi (SCP) per controllare quale modalità di uscita i chiamanti possono specificare quando creano o aggiornano i cluster.

La chiave condizionale si applica alle seguenti azioni:

- `eks:CreateCluster`
- `eks:UpdateClusterConfig`

Ad esempio, il seguente SCP nega la creazione e gli aggiornamenti della configurazione del cluster a meno che il chiamante non specifichi: `CUSTOMER_ROUTED`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireCustomerRoutedControlPlane",
      "Effect": "Deny",
      "Action": [
        "eks:CreateCluster",
        "eks:UpdateClusterConfig"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "StringNotEquals": {
          "eks:controlPlaneEgressMode": "CUSTOMER_ROUTED"
        }
      }
    }
  ]
}
```

Utilizza questa policy per far sì che tutti i cluster nuovi e aggiornati dell'organizzazione utilizzino la modalità di uscita. `CUSTOMER_ROUTED`

## Configurazione del provider OIDC

Se il cluster utilizza un provider di identità OIDC, il piano di controllo deve essere in grado di raggiungere l'endpoint di rilevamento OIDC tramite HTTPS (porta 443). Questo vale per i ruoli IAM per gli account di servizio o per un provider di identità OIDC associato per l'autenticazione del cluster. Non è prevista alcuna OIDC-specific impostazione; utilizza lo stesso percorso di uscita configurato in Prerequisiti. ??? Per consentirlo:

1. Verifica che le sottoreti del piano di controllo abbiano un percorso che copra l'endpoint OIDC (in genere un percorso predefinito verso il dispositivo di uscita, come un gateway NAT).
2. Verifica che il gruppo di sicurezza del cluster consenta il protocollo TCP 443 in uscita.
3. Verifica che i NACL della sottorete consentano il traffico TCP 443 in uscita e il traffico effimero di ritorno in entrata (porte 1024—65535).

L'endpoint dipende dal tuo provider:

- Provider OIDC Amazon EKS (predefinito): `oidc.eks.region-code.amazonaws.com`
- Provider OIDC personalizzato: l'URL dell'emittente che hai configurato.

Se l'autenticazione OIDC fallisce, per la procedura di risoluzione dei problemi, consulta. the section called "Provider OIDC non raggiungibile"

## Verifica la connettività

Dopo aver configurato l'`CUSTOMER_ROUTED`uscita, verifica che il piano di controllo possa raggiungere le tue risorse VPC:

1. Controlla la modalità di uscita corrente: conferma che il cluster stia utilizzando la modalità prevista.

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.resourcesVpcConfig.controlPlaneEgressMode" \
    --region region-code
```

2. Controlla lo stato del cluster: il cluster deve essere in `ACTIVE` stato.

```
aws eks describe-cluster --name my-cluster --query "cluster.status" --region region-
code
```

3. Verifica la connettività del webhook: se hai configurato i webhook di ammissione, crea una risorsa che attivi il webhook e confermi che ha esito positivo.
4. Verifica la registrazione del nodo: avvia un nodo e conferma che si unisca correttamente al cluster.

```
kubectl get nodes
```

5. Verifica l'OIDC: se utilizzi i ruoli IAM per gli account di servizio (IRSA), verifica che i pod possano assumere i propri ruoli IAM.

Per la risoluzione dei problemi più comuni, consulta. the section called “Risolvi i problemi relativi all'uscita”

# Modifica questa pagina su GitHub

## Risoluzione dei problemi relativi all'uscita del piano di controllo

Quando si utilizza la modalità di uscita del piano `CUSTOMER_ROUTED` di controllo, sei responsabile della connettività di rete dal piano di controllo ENI. Questa pagina descrive i problemi più comuni e le relative soluzioni.

### Rileva un webhook difettoso

Quando il piano di controllo non riesce a raggiungere un server webhook o un provider OIDC, il sintomo di solito si manifesta come un timeout del webhook. Per confermare, creare o modificare una risorsa che attiva il webhook e verificare l'errore:

```
kubectl apply -f my-resource.yaml
```

Un errore di connettività o DNS restituisce in genere un errore simile al seguente:

```
Error from server (InternalError): error when creating "my-resource.yaml": Internal
 error occurred:
failed calling webhook "my-webhook.example.com": failed to call webhook:
Post "https://my-webhook.example.com/validate?timeout=10s": context deadline exceeded
```

Puoi anche controllare gli eventi recenti per verificare la presenza di errori nei webhook nel cluster:

```
kubectl get events --all-namespaces --field-selector reason=FailedCreate
```

- Se l'errore è un timeout (`context deadline exceeded`) o una connessione rifiutata, il piano di controllo non può raggiungere l'endpoint del webhook. Consulta le sezioni the section called "Nessuna via di uscita verso gli endpoint richiesti", the section called "I NACL bloccano il webhook o controllano il traffico aereo" e the section called "Gruppi di sicurezza che impediscono l'accesso".
- Se l'errore menziona un errore DNS o nessun errore simile sull'host, il piano di controllo non è in grado di risolvere l'endpoint. Per informazioni, consulta the section called "Errore di aggiornamento del set di opzioni DHCP".

## Nessuna via di uscita verso gli endpoint richiesti

Caratteristiche:

- Timeout dei webhook di ammissione.
- L'individuazione del provider OIDC non riesce.
- La creazione o l'aggiornamento dei cluster si bloccano.

Causa:

Le sottoreti dell'interfaccia di rete del piano di controllo non hanno un percorso funzionante verso gli endpoint che il piano di controllo deve raggiungere. Nella maggior parte dei casi, nella tabella delle rotte delle sottoreti manca una route predefinita verso un dispositivo di uscita. In alternativa, quel dispositivo non è configurato correttamente. Il dispositivo di uscita è in genere un gateway NAT. Tuttavia, può essere un'istanza NAT, un firewall o un dispositivo proxy o un gateway di transito verso un VPC in uscita centralizzato.

Soluzione::

1. Identifica le sottoreti utilizzate dal cluster per le interfacce di rete del piano di controllo:

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.resourcesVpcConfig.subnetIds"
```

2. Per ogni sottorete, controllate la tabella di routing associata:

```
aws ec2 describe-route-tables \
    --filters "Name=association.subnet-id,Values=subnet-ExampleID1"
```

3. Verifica che esista un percorso per `0.0.0.0/0` (o un percorso che copra l'endpoint) che punta al tuo dispositivo di uscita. Se manca, aggiungi il percorso. L'esempio seguente aggiunge una route di gateway NAT; sostituite la vostra destinazione di uscita (ad esempio, un gateway di transito o un'interfaccia di rete):

```
aws ec2 create-route \
    --route-table-id rtb-ExampleID \
    --destination-cidr-block 0.0.0.0/0 \
    --nat-gateway-id nat-ExampleID
```

## I NACL bloccano il webhook o controllano il traffico aereo

Caratteristiche:

- Timeout delle chiamate al webhook di ammissione (errore:). `failed calling webhook`
- Errori intermittenti durante la creazione o la modifica di risorse Kubernetes che utilizzano webhook mutanti o di convalida.

Causa:

Gli ACL di rete sul piano di controllo, le sottoreti ENI bloccano il traffico in uscita verso gli endpoint webhook o bloccano il traffico effimero di ritorno alle porte in entrata.

Soluzione::

1. Identifica i NACL associati alle sottoreti del tuo piano di controllo:

```
aws ec2 describe-network-acls \
    --filters "Name=association.subnet-id,Values=subnet-ExampleID1"
```

2. Assicuratevi che esistano le seguenti regole:

| Direzione | Protocollo | Intervallo porte | Destination/ Source | Azione |
|---|---|---|---|---|
| In uscita | TCP | 443 | 0.0.0. 0/0 (o webhook CIDR) | Consenso |
| In uscita | TCP | 10250 | CIDR VPC | Consenso |
| In entrata | TCP | 1024—65535 | 0,0,0. 0/0 | Consenti (traffico temporaneo di ritorno) |

> **Note**
>
> I NACL sono apolidi. È necessario consentire esplicitamente il traffico di ritorno sulle porte temporanee (1024—65535) nelle regole in entrata.
> Queste regole riguardano due percorsi diversi. La regola della porta 443 riguarda il traffico in uscita verso gli endpoint webhook e OIDC, che lascia il VPC attraverso il dispositivo in uscita. La regola della porta 10250 è per l'API kubelet, che rimane all'interno del tuo VPC tra il piano di controllo e i nodi. Un dispositivo di uscita mancante non influisce sulla porta 10250, ma un ACL di rete restrittivo può bloccarlo.

## Gruppi di sicurezza che impediscono l'accesso

Caratteristiche:

- Le chiamate Webhook hanno esito negativo.
- Il piano di controllo non può raggiungere l'API kubelet sui nodi (porta 10250).
- `kubectl exec`, o `kubectl logs` fallire. `kubectl port-forward`

Causa:

Il gruppo di sicurezza collegato al piano di controllo ENiS (il gruppo di sicurezza del cluster) non consente il traffico in uscita sulle porte richieste.

Soluzione::

1. Identifica il gruppo di sicurezza del cluster:

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.resourcesVpcConfig.clusterSecurityGroupId"
```

2. Verifica che le regole in uscita consentano:

| Protocollo | Porta | Destinazione |
|---|---|---|
| TCP | 443 | 0.0.0. 0/0 (endpoint webhook, provider OIDC) |
| TCP | 10250 | Gruppo di sicurezza del nodo o VPC CIDR (API kubelet) |

3. Se le regole in uscita sono restrittive, aggiungi regole per il traffico richiesto:

```
aws ec2 authorize-security-group-egress \
    --group-id sg-ExampleClusterSG \
    --protocol tcp \
    --port 443 \
    --cidr 0.0.0.0/0
```

> **Note**
>
> Se hai requisiti di uscita rigorosi e conosci gli intervalli IP del tuo webhook e degli endpoint OIDC, puoi applicare la regola della porta 443 a quei CIDR specifici anziché a. `0.0.0.0/0` La regola della porta 10250 (API kubelet) è VPC-internal: applicala al gruppo di sicurezza del nodo o al CIDR VPC anziché a Internet.

## Errore di aggiornamento del set di opzioni DHCP

Caratteristiche:

- La risoluzione DNS non riesce dal piano di controllo.

- Le operazioni del cluster che richiedono ricerche DNS (rilevamento OIDC, risoluzione webhook) hanno esito negativo.
- Il problema si verifica dopo la modifica delle opzioni DHCP del VPC o dopo un aggiornamento del piano di controllo.

Causa:

Il set di opzioni DHCP VPC è stato modificato. In alternativa, non include `AmazonProvidedDNS` nei suoi server di nomi di dominio. Potrebbe anche mancare un altro resolver in grado di risolvere i nomi necessari al piano di controllo. Il piano di controllo rileva automaticamente le modifiche al set di opzioni DHCP e applica le nuove impostazioni DNS, in genere entro un'ora. Il piano di controllo può eseguire questa operazione solo quando il ruolo IAM del cluster concede le autorizzazioni di lettura richieste di Amazon EC2.

Soluzione::

1. Verifica l'impostazione delle opzioni DHCP per il tuo VPC:

```
aws ec2 describe-vpcs --vpc-ids vpc-ExampleID \
    --query "Vpcs[0].DhcpOptionsId" \
    --region region-code
```

```
aws ec2 describe-dhcp-options --dhcp-options-ids dopt-ExampleID --region region-code
```

2. Verifica che `domain-name-servers` includa `AmazonProvidedDNS` (il resolver Amazon-provided DNS, che è la base del tuo VPC IPv4 CIDR più due) o un altro resolver in grado di risolvere i nomi necessari al piano di controllo.
3. Conferma le concessioni `ec2:DescribeVpcs` del ruolo IAM del cluster e. `ec2:DescribeDhcpOptions` Senza queste autorizzazioni, il piano di controllo non può leggere le opzioni DHCP aggiornate e non può aggiornare le impostazioni DNS. Per ulteriori informazioni, consulta il ruolo IAM del cluster Amazon EKS.
4. Dopo una modifica delle opzioni DHCP, attendi fino a un'ora affinché il piano di controllo rilevi e applichi automaticamente le nuove impostazioni. Non è richiesto alcun aggiornamento del cluster o la sostituzione dell'istanza. Se la risoluzione DNS continua a fallire dopo un'ora e le autorizzazioni di cui sopra sono disponibili, contatta l'assistenza AWS .

## Problemi di routing IPv6

Caratteristiche:

- I cluster IPv6 non possono raggiungere endpoint OIDC o webhook esterni.
- La registrazione dei nodi funziona su IPv4 ma i servizi IPv6 falliscono.

Causa:

Nella tabella di routing delle sottoreti manca una `::/0` route verso un gateway Internet solo in uscita oppure la sicurezza non consente il traffico IPv6. groups/NACLs

Soluzione::

1. Verifica che esista un gateway Internet solo in uscita e che sia collegato al VPC:

```
aws ec2 describe-egress-only-internet-gateways \
    --filters "Name=attachment.vpc-id,Values=vpc-ExampleID"
```

2. Verificate che la tabella dei percorsi per le sottoreti del piano di controllo abbia un percorso: `::/0`

```
aws ec2 describe-route-tables \
    --filters "Name=association.subnet-id,Values=subnet-ExampleID1" \
    --query "RouteTables[0].Routes[?DestinationIpv6CidrBlock=='::/0']"
```

3. Se manca, aggiungi il percorso:

```
aws ec2 create-route \
    --route-table-id rtb-ExampleID \
    --destination-ipv6-cidr-block ::/0 \
    --egress-only-internet-gateway-id eigw-ExampleID
```

4. Assicurati che i NAC e i gruppi di sicurezza consentano l'IPv6 in uscita sulla porta 443 e le porte effimere in entrata.

## Provider OIDC non raggiungibile

Caratteristiche:

- `IAM roles for service accounts`(IRSA) fallisce: i pod non possono assumere ruoli.

- Gli eventi del cluster mostrano errori di rilevamento OIDC.

Causa:

Il piano di controllo non può raggiungere l'endpoint del provider OIDC (ad esempio`oidc.eks.region-code.amazonaws.com`) perché l'uscita è bloccata.

Soluzione::

1. Verificate che il percorso di uscita e la tabella delle rotte consentano il traffico HTTPS in uscita. Per la procedura di risoluzione dei problemi in caso di mancata o errata configurazione del percorso di uscita, consulta. the section called "Nessuna via di uscita verso gli endpoint richiesti"
2. Verificate che il gruppo di sicurezza del cluster consenta il protocollo TCP 443 in uscita di (vedere). `0.0.0.0/0` the section called "Gruppi di sicurezza che impediscono l'accesso"

# Modifica questa pagina su GitHub

# Dimensionamento dell'elaborazione dei cluster con Karpenter e Cluster Autoscaler

Il dimensionamento automatico è una funzione che dimensiona automaticamente in verticale e in orizzontale le risorse per soddisfare le esigenze in continua evoluzione. Questa è una delle principali funzioni Kubernetes, la cui esecuzione manuale richiederebbe altrimenti diverse risorse umane.

## EKS Auto Mode

EKS Auto Mode di Amazon dimensiona automaticamente le risorse di calcolo del cluster. Se un pod non può adattarsi ai nodi esistenti, EKS Auto Mode ne crea uno nuovo. EKS Auto Mode consolida anche i carichi di lavoro ed elimina i nodi. EKS Auto Mode si basa su Karpenter.

Per ulteriori informazioni, consultare:

- Modalità automatica EKS
- the section called "Creazione di pool di nodi"
- the section called "Implementazione di un carico di lavoro di decompressione"

# Ulteriori soluzioni

Amazon EKS supporta due ulteriori prodotti con dimensionamento automatico:

## Karpenter

Karpenter è un Cluster Autoscaler Kubernetes flessibile e ad alte prestazioni che aiuta a migliorare la disponibilità delle applicazioni e l'efficienza del cluster. Karpenter avvia risorse di calcolo di dimensioni corrette, ad esempio le istanze di Amazon EC2, in risposta alla modifica del carico dell'applicazione in meno di un minuto. Integrando Kubernetes con AWS, Karpenter è in grado di effettuare il provisioning di risorse di calcolo just-in-time che soddisfano con precisione i requisiti del carico di lavoro. Karpenter effettua automaticamente il provisioning di nuove risorse di calcolo in base ai requisiti specifici dei carichi di lavoro del cluster. Questi includono requisiti di calcolo, archiviazione, accelerazione e pianificazione. Amazon EKS supporta i cluster che utilizzano Karpenter, anche se Karpenter funziona con qualsiasi cluster di Kubernetes conforme. Per ulteriori informazioni, consulta la documentazione di Karpenter.

> **Important**
>
> Karpenter è un software open source che i clienti AWS sono tenuti a installare, configurare e gestire nei propri cluster Kubernetes. AWS fornisce supporto tecnico quando Karpenter viene eseguito senza modifiche usando una versione compatibile nei cluster Amazon EKS. È fondamentale che i clienti mantengano la disponibilità e la sicurezza del controller Karpenter, nonché adeguate procedure di test quando lo aggiornano o aggiornano il cluster Kubernetes in cui è in esecuzione, proprio come qualsiasi altro software gestito dal cliente. Non esiste un accordo sul livello di servizio (SLA) AWS per Karpenter e i clienti sono responsabili di garantire che le istanze EC2 avviate da Karpenter soddisfino i loro requisiti aziendali.

## Cluster Autoscaler

Il Cluster Autoscaler Kubernetes regola automaticamente il numero di nodi nel cluster quando i pod non riescono o vengono riprogrammati su altri nodi. Il Cluster Autoscaler utilizza i gruppi con scalabilità automatica. Per ulteriori informazioni consultare Cluster Autoscaler su AWS.

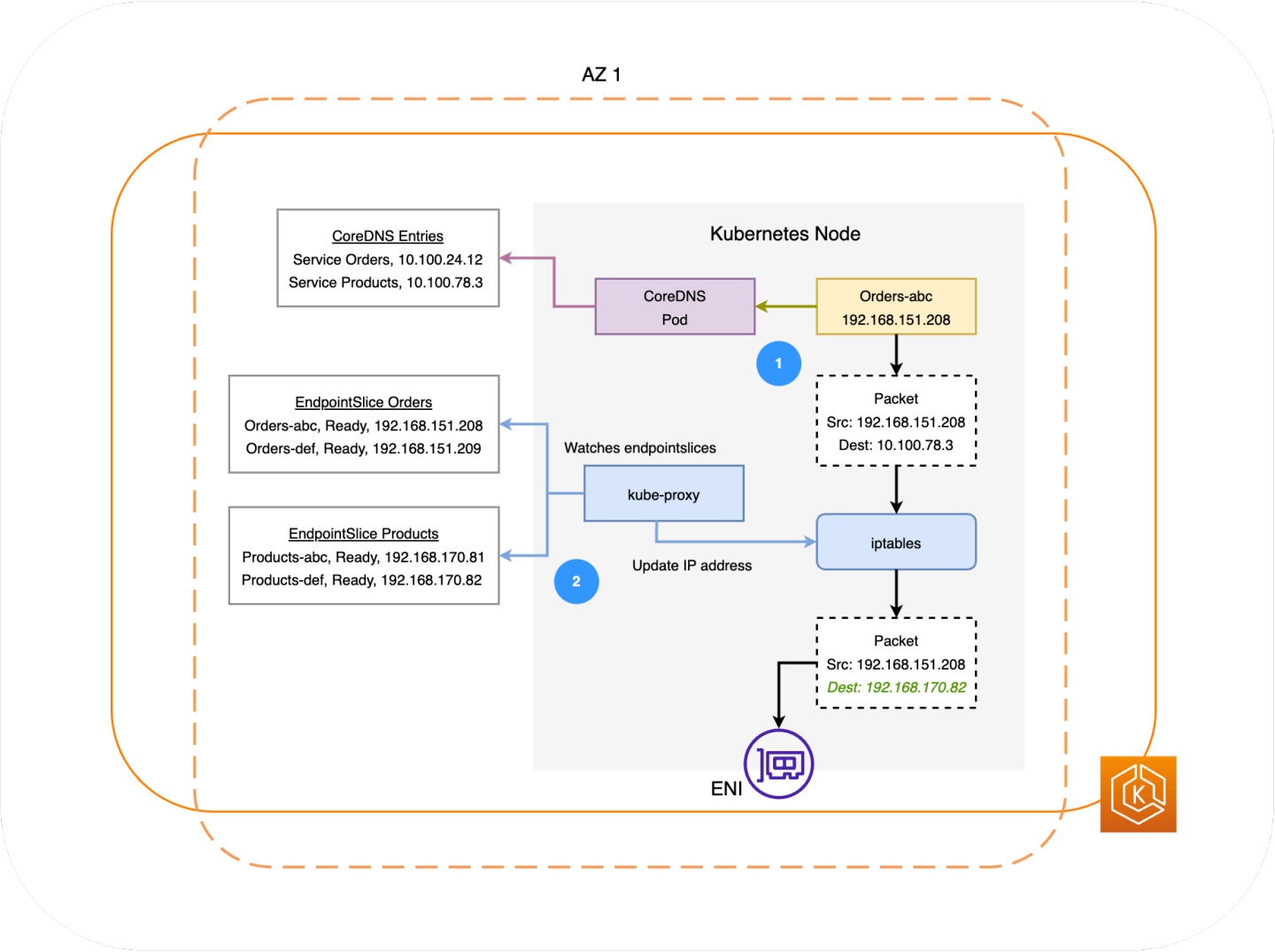
# Ulteriori informazioni relative allo spostamento zonale in controller di ripristino delle applicazioni Amazon (ARC)

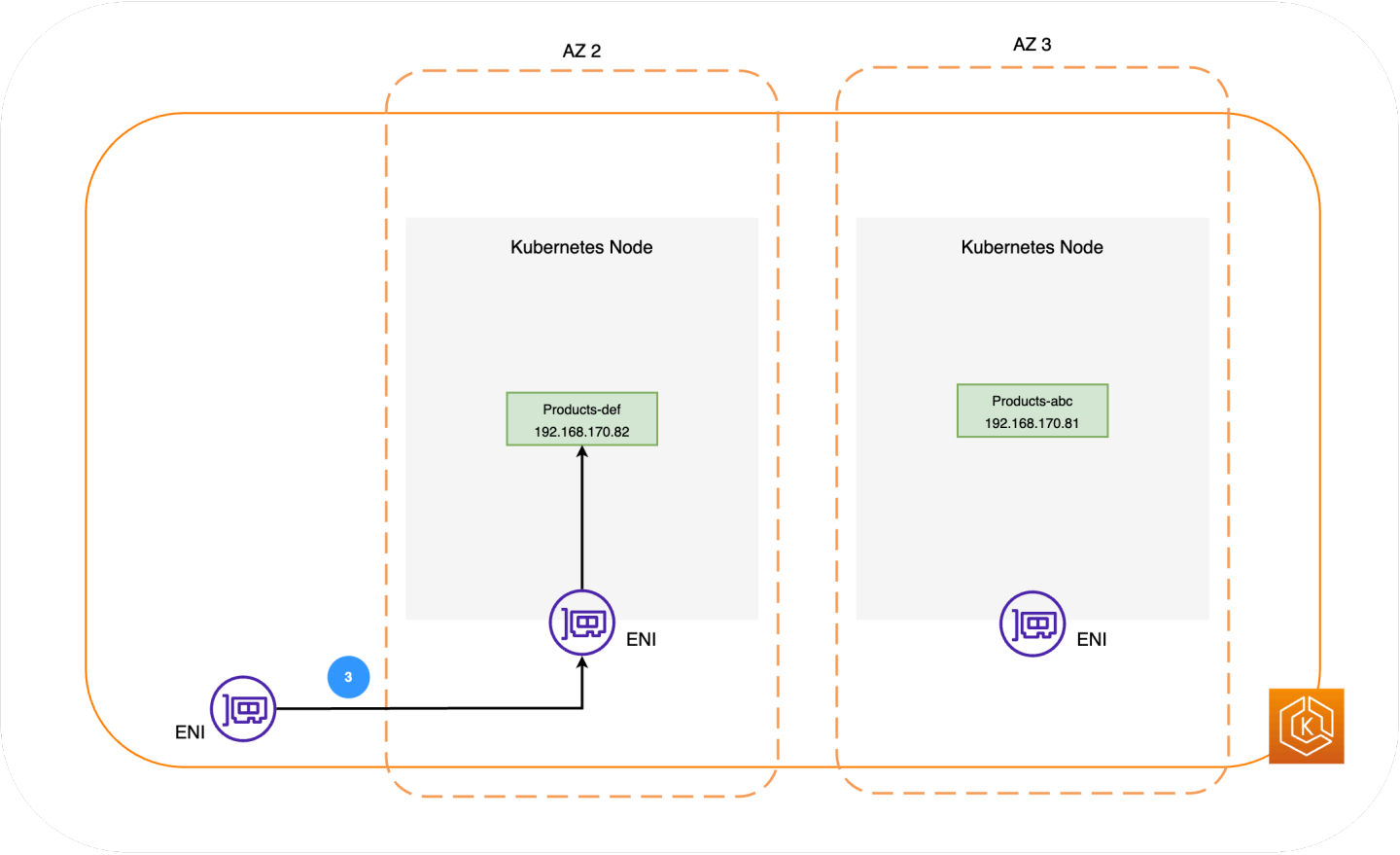
Kubernetes dispone di funzionalità native che consentono di rendere le applicazioni più resistenti a eventi come il deterioramento dello stato di integrità o il deterioramento di una zona di disponibilità (AZ). Quando esegui i carichi di lavoro in un cluster Amazon EKS, è possibile migliorare ulteriormente la tolleranza ai guasti e il ripristino delle applicazioni dell'ambiente applicativo utilizzando lo spostamento zonale o lo spostamento zonale automatico del controller di ripristino delle applicazioni Amazon (ARC). Lo spostamento zonale ARC è progettato per essere una misura temporanea che consente di spostare il traffico di una risorsa lontano da una AZ interessata fino alla scadenza dello spostamento zonale o all'annullamento dello stesso. È possibile estendere lo spostamento zonale, se necessario.

Puoi avviare uno spostamento zonale per un cluster EKS oppure puoi consentire AWS lo spostamento del traffico per te abilitando lo spostamento automatico zonale. Questo spostamento aggiorna il flusso di traffico di rete da est a ovest nel cluster per considerare solo gli endpoint di rete per i pod in esecuzione su nodi worker in AZ integre. Inoltre, qualsiasi ALB o NLB che gestisce il traffico in ingresso per le applicazioni nel cluster EKS indirizzerà automaticamente il traffico verso le destinazioni nelle AZ integre. Per i clienti che cercano gli obiettivi di disponibilità più elevati, nel caso in cui una zona AZ sia compromessa, può essere importante consentire di indirizzare tutto il traffico lontano dalla AZ compromessa fino a quando non si ripristina. Per questo, puoi anche abilitare ALB o NLB con spostamento zonale ARC.

## Comprensione del flusso di traffico di rete est-ovest tra i pod

Il diagramma seguente illustra due esempi di carichi di lavoro, Ordini e Prodotti. Lo scopo di questo esempio è mostrare come comunicano carichi di lavoro e Pod in diverse AZ.


AZ 1
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
Kubernetes Node
CoreDNS
Pod
Orders-abc
192.168.151.208
1
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
iptables
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
2
Update IP address
Packet
Src: 192.168.151.208
Dest: 192.168.170.82
ENI




1. Affinché gli Ordini comunichino con i Prodotti, gli Ordini devono prima risolvere il nome DNS del servizio di destinazione. Ordini comunica con CoredNS per recuperare l'indirizzo IP virtuale (IP cluster) per quel servizio. Dopo che Ordini ha risolto il nome del servizio Prodotti, invia il traffico a quell'indirizzo IP di destinazione.
2. Il kube-proxy viene eseguito su ogni nodo del cluster e controlla continuamente i servizi. EndpointSlices Quando viene creato un servizio, ne EndpointSlice viene creato e gestito in background dal controller. EndpointSlice Ognuno EndpointSlice ha un elenco o una tabella di endpoint che contiene un sottoinsieme di indirizzi Pod, insieme ai nodi su cui sono in esecuzione. Il kube-proxy imposta le regole di routing per ciascuno di questi endpoint di pod utilizzando `iptables` sui nodi. Il kube-proxy è anche responsabile di una forma base di bilanciamento del carico, che consiste nel reindirizzare il traffico destinato all'indirizzo IP del cluster di un servizio per inviarlo invece direttamente all'indirizzo IP di un pod. Il kube-proxy esegue questa operazione riscrivendo l'indirizzo IP di destinazione sulla connessione in uscita.
3. Quindi, i pacchetti di rete sono inviati al pod dei prodotti in AZ 2 utilizzando ENI sui rispettivi nodi, come mostrato nel diagramma precedente.

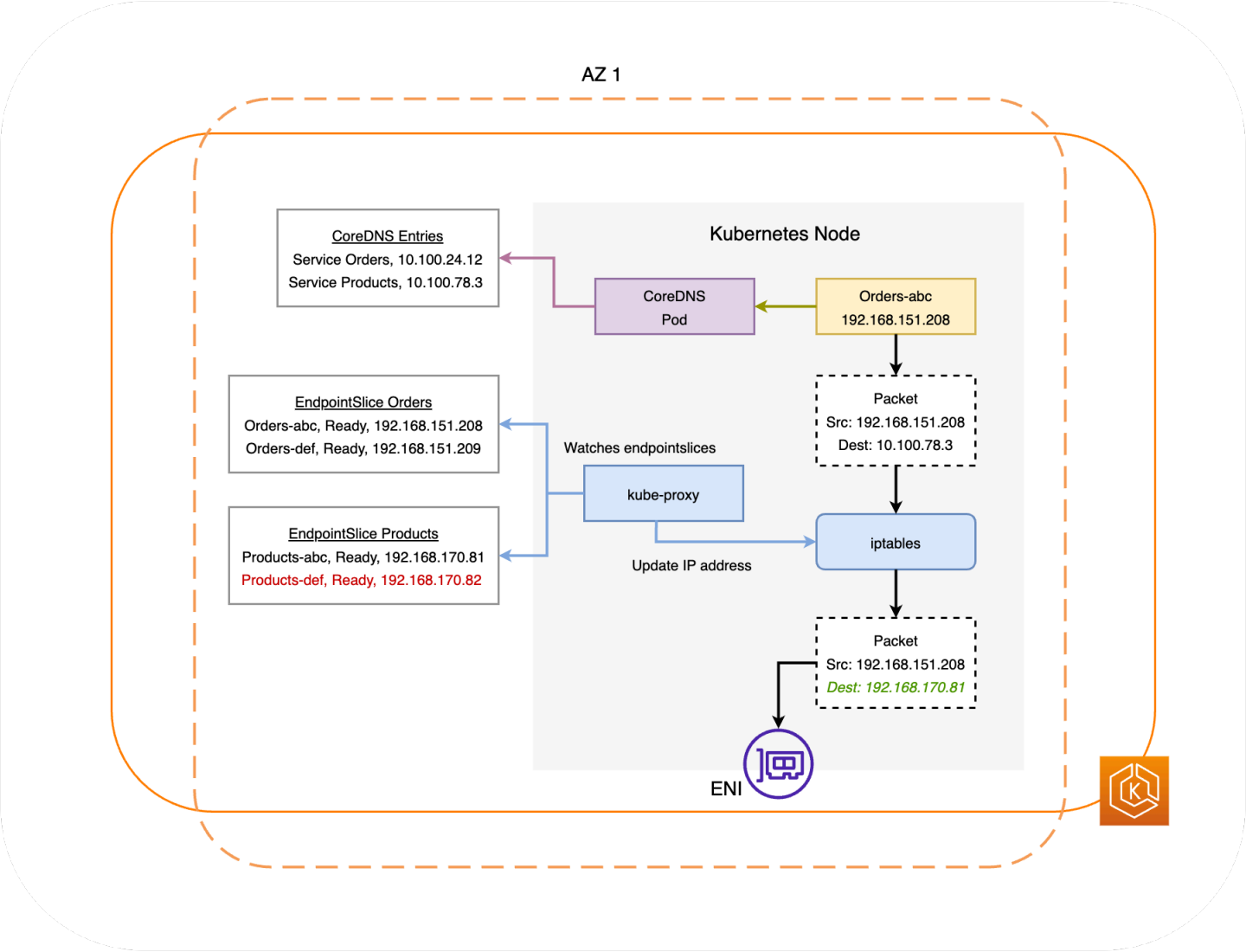
## Comprensione dello spostamento zonale ARC in Amazon EKS

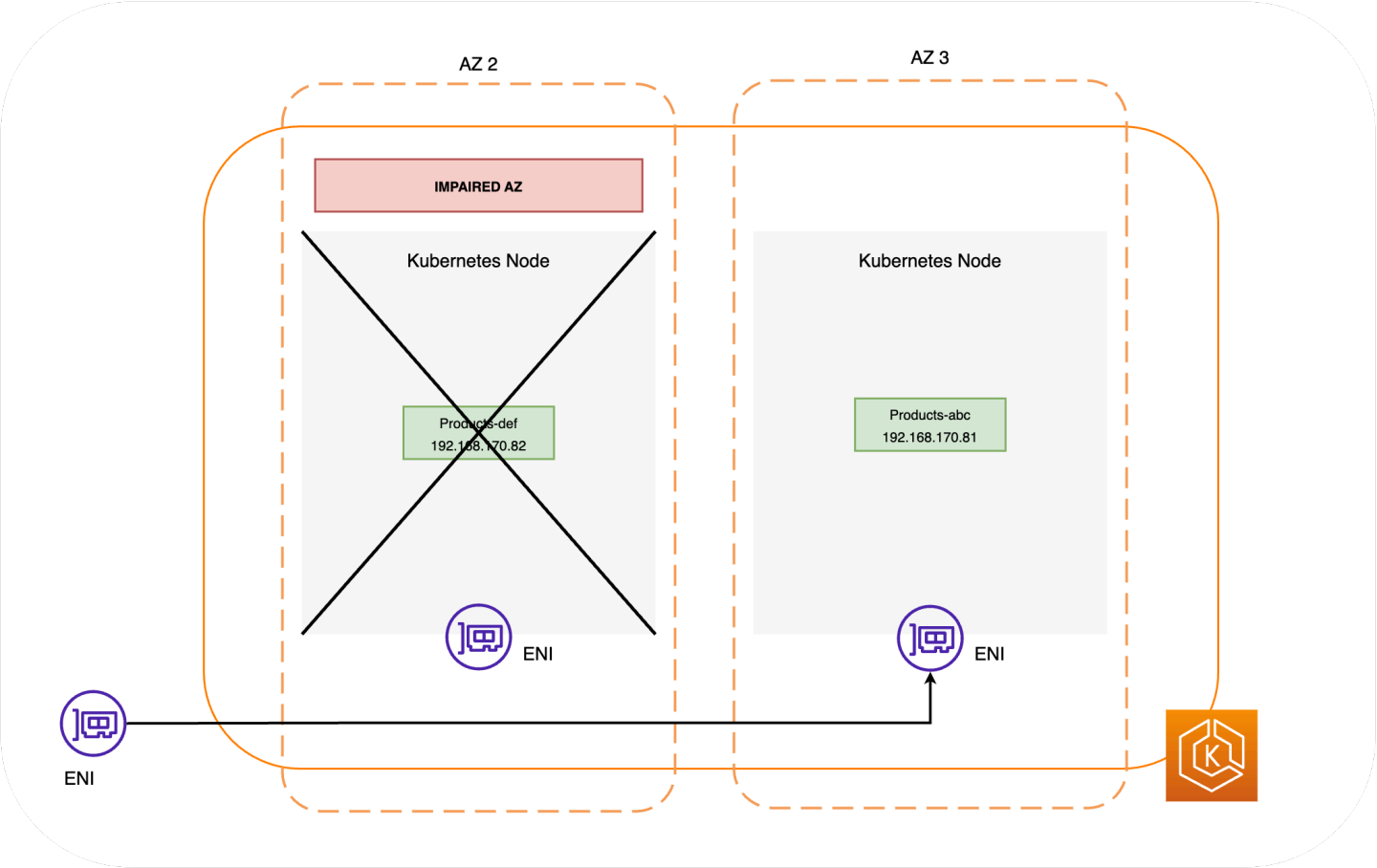
Se il tuo ambiente presenta un deterioramento di AZ, è possibile avviare uno spostamento zonale per l'ambiente del cluster EKS. In alternativa, puoi consentire AWS la gestione dello spostamento del traffico per te con lo spostamento automatico zonale. Con l'autoshift zonale, AWS monitora lo stato generale dell'AZ e risponde a un potenziale deterioramento dell'AZ spostando automaticamente il traffico dall'AZ compromesso nell'ambiente del cluster.

Dopo che il tuo cluster Amazon EKS ha abilitato lo spostamento zonale con ARC, puoi avviare uno spostamento zonale o abilitare lo spostamento automatico zonale utilizzando la console ARC, la CLI o le API zonal shift e zonal autoshift. AWS Durante uno spostamento zonale EKS, è eseguito automaticamente quanto segue:

- Tutti i nodi della zona AZ interessata sono isolati. Ciò impedisce al sistema di pianificazione Kubernetes di pianificare nuovi pod su nodi in una zona AZ non integra.
- Se utilizzi EKS Auto Mode, EKS Auto Mode interrompe automaticamente il provisioning di nuovi nodi nell'AZ compromesso. L'EKS Auto Mode sospende inoltre le azioni volontarie di interruzione, come il consolidamento e la deriva, che potrebbero influire sull'AZ compromesso, impedendo così la chiusura volontaria dell'istanza nell'AZ compromesso per prevenire tempeste di tentativi. I pod con requisiti di pianificazione rigorosi destinati all'AZ compromesso (come associazioni di volume persistenti, affinità tra nodi o rigidi vincoli di diffusione della topologia) non comportano tentativi di lancio di nuovi nodi verso tale AZ.
- Se utilizzi i gruppi di nodi gestiti, il ribilanciamento delle zone di disponibilità è sospeso e il gruppo Auto Scaling è aggiornato per garantire che i nuovi nodi del piano dati EKS siano lanciati solo in AZ integre.
- I nodi all'interno di AZ non integra non sono terminati e i pod non sono espulsi dai nodi. In questo modo, quando uno spostamento zonale scade o è annullato, il traffico può essere restituito in tutta sicurezza ad AZ per sfruttarne la piena capacità.
- Il EndpointSlice controller trova tutti gli endpoint del Pod nell'AZ compromesso e li rimuove da quelli pertinenti. EndpointSlices Ciò garantisce che solo gli endpoint dei pod in AZ integre siano destinati a ricevere il traffico di rete. Quando uno spostamento zonale viene annullato o scade, il EndpointSlice controller lo aggiorna EndpointSlices per includere gli endpoint nell'AZ ripristinato.

I seguenti diagrammi forniscono una panoramica di alto livello su come uno spostamento zonale EKS garantisce che solo gli endpoint dei pod integri siano presi di mira nell'ambiente del cluster.


AZ 1
Kubernetes Node
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
CoreDNS
Pod
Orders-abc
192.168.151.208
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
Update IP address
iptables
Packet
Src: 192.168.151.208
Dest: 192.168.170.81
ENI




# Requisiti per lo spostamento zonale EKS

> **Important**
>
> Uno spostamento zonale sposta tutto il traffico all'interno del cluster da una zona di disponibilità. Se i carichi di lavoro non sono distribuiti su più AZ con repliche sufficienti, l'avvio di uno spostamento zonale può di per sé causare un impatto sulla disponibilità delle applicazioni perché non ci sono endpoint integri per ricevere traffico. È necessario verificare che l'ambiente cluster possa funzionare con un AZ in meno prima di abilitare lo spostamento automatico zonale o avviare uno spostamento zonale manuale.

Affinché lo spostamento zonale funzioni correttamente con EKS, è necessario configurare in anticipo l'ambiente del cluster in modo che sia resiliente a una compromissione di AZ. Di seguito è riportato un elenco di opzioni di configurazione che aiutano a garantire la resilienza.

- Provisioning dei nodi worker del cluster su più AZ

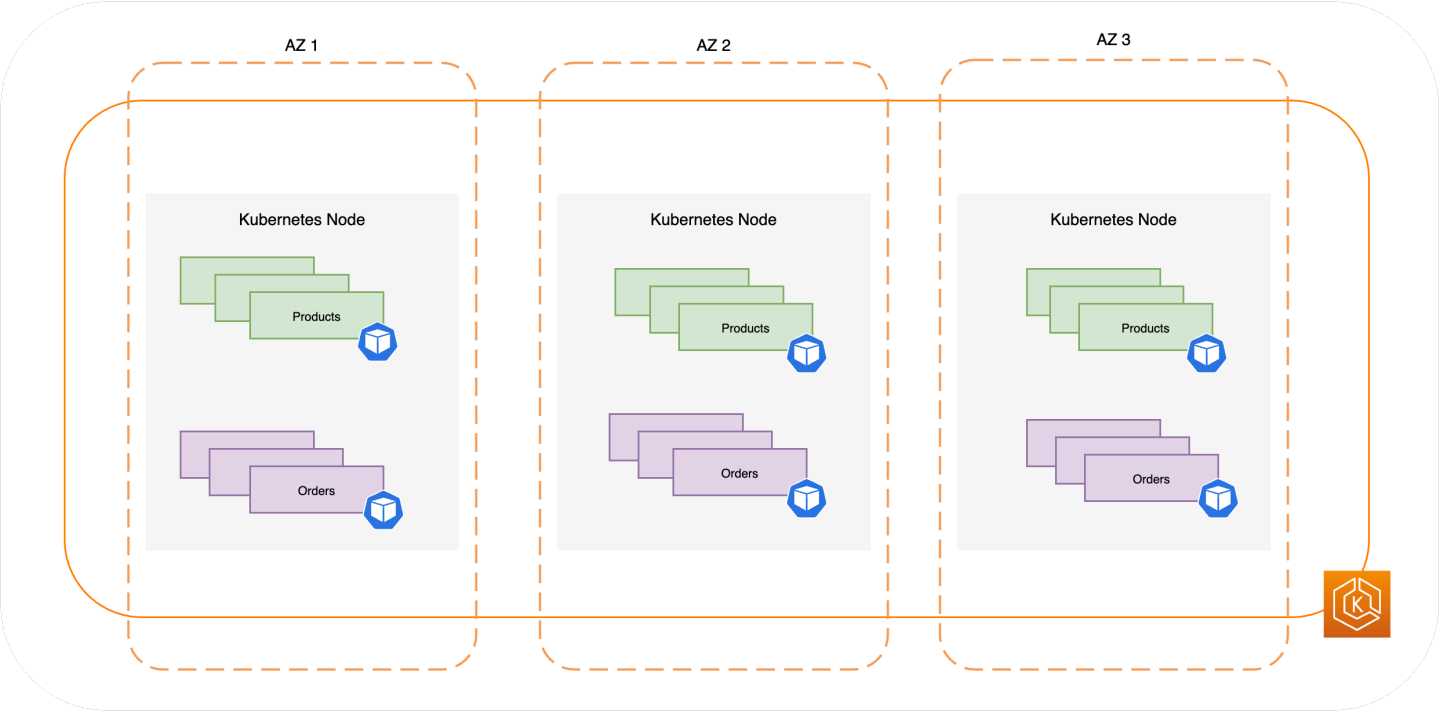
- Consulta a una capacità di elaborazione sufficiente per consentire la rimozione di una singola AZ
- Pre-scale i tuoi Pod, incluso CoreDNS, in ogni AZ
- Distribuisci più repliche di pod su tutte le AZ, per assicurarti che, quando ti sposti da una sola AZ, avrai comunque una capacità sufficiente
- Effettua la co-locazione di pod interdipendenti o correlati nella stessa AZ
- Verifica che il tuo ambiente cluster funzioni come previsto senza una AZ avviando manualmente uno spostamento zonale dalla zona AZ. In alternativa, è possibile abilitare lo spostamento zonale automatico e fare affidamento sulle esecuzioni pratiche di spostamento automatico. I test con spostamenti zonali manuali o pratici non sono necessari affinché lo spostamento zonale funzioni in EKS, ma sono fortemente consigliati.

## Esegui il provisioning dei nodi workerEKS in più zone di disponibilità

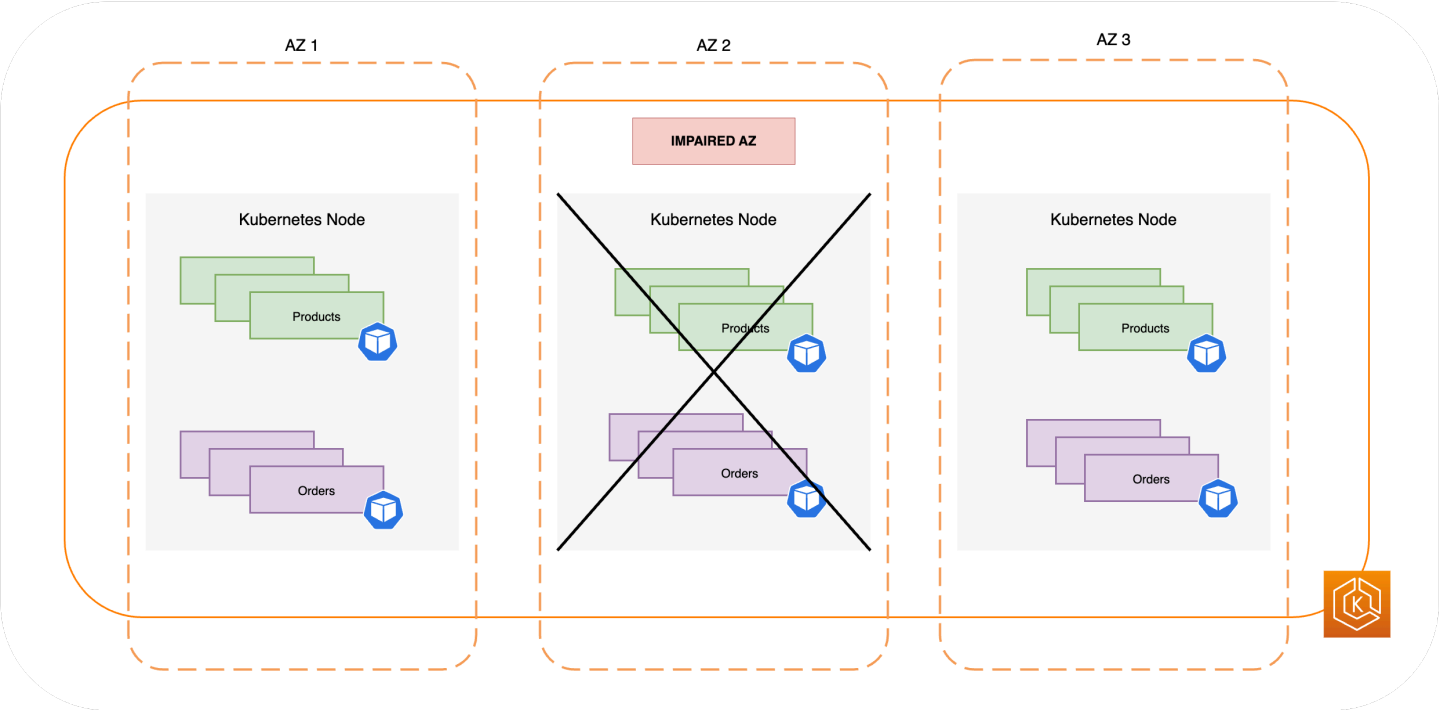
AWS Le regioni hanno più sedi separate con data center fisici, note come zone di disponibilità (AZ). Le AZ sono progettate per essere isolate fisicamente l'una dall'altra per evitare impatti simultanei che potrebbero interessare un'intera regione. Quando si effettua il provisioning di un cluster EKS, si consiglia di distribuire i nodi di lavoro su più AZ in una regione. Ciò contribuisce a rendere l'ambiente cluster più resiliente alla compromissione di una singola AZ e consente di mantenere un'elevata disponibilità per le applicazioni eseguite nelle altre AZ. Quando avvii un passaggio zonale dall'AZ interessato, la rete interna al cluster dell'ambiente EKS si aggiorna automaticamente per utilizzare solo AZ integre, per contribuire a mantenere un'elevata disponibilità del cluster.

La garanzia di disporre di una configurazione multi-AZ per l'ambiente EKS migliora l'affidabilità complessiva del sistema. Tuttavia, gli ambienti multi-AZ influenzano il modo in cui i dati delle applicazioni sono trasferiti ed elaborati, il che a sua volta ha un impatto sui costi di rete dell'ambiente. In particolare, il traffico in uscita frequente tra le zone (traffico distribuito tra le AZ) può avere un impatto importante sui costi relativi alla rete. È possibile applicare diverse strategie per controllare la quantità di traffico tra le zone tra i pod nel cluster EKS e ridurre i costi associati. Per ulteriori informazioni su come ottimizzare i costi di rete durante l'esecuzione di ambienti EKS ad alta disponibilità, consulta queste best practice.

Il diagramma seguente illustra un ambiente EKS ad alta disponibilità con tre AZ integre.




Il diagramma seguente illustra come un ambiente EKS con tre AZ sia resiliente a una compromissione di AZ e rimanga altamente disponibile perché rimangono due AZ integre.



## Consulta a una capacità di elaborazione sufficiente per resistere alla rimozione di una singola zona di disponibilità

Per ottimizzare l'utilizzo delle risorse e i costi per l'infrastruttura di elaborazione nel piano dati EKS, è consigliabile allineare la capacità di elaborazione ai requisiti del carico di lavoro. Tuttavia, se tutti i nodi worker sono a piena capacità, è necessario aggiungere nuovi nodi worker al piano dati EKS prima di poter pianificare nuovi pod. Quando si eseguono carichi di lavoro critici, in genere è buona norma impiegare una capacità ridondante online per gestire scenari quali aumenti improvvisi del carico e problemi di integrità dei nodi. Se prevedi di utilizzare lo spostamento zonale, intendi rimuovere un'intera AZ di capacità in caso di compromissione. Ciò significa che è necessario regolare la capacità di elaborazione ridondante in modo che sia sufficiente a gestire il carico anche con una delle AZ offline.

Quando si ridimensionano le risorse di elaborazione, il processo di aggiunta di nuovi nodi al piano dati EKS richiede del tempo. Ciò può avere implicazioni sulle prestazioni e sulla disponibilità in tempo reale delle applicazioni, soprattutto in caso di una compromissione zonale. Il tuo ambiente EKS dovrebbe essere in grado di assorbire il peso della perdita di una AZ senza compromettere l'esperienza degli utenti finali o dei clienti. Ciò significa ridurre al minimo o eliminare il ritardo tra il momento in cui è necessario un nuovo pod e quello in cui è effettivamente pianificato su un nodo worker.

Inoltre, in caso di compromissione zonale, dovresti cercare di mitigare il rischio di incorrere in un limite di capacità di elaborazione che impedirebbe l'aggiunta di nuovi nodi necessari al piano dati EKS nelle AZ integre.

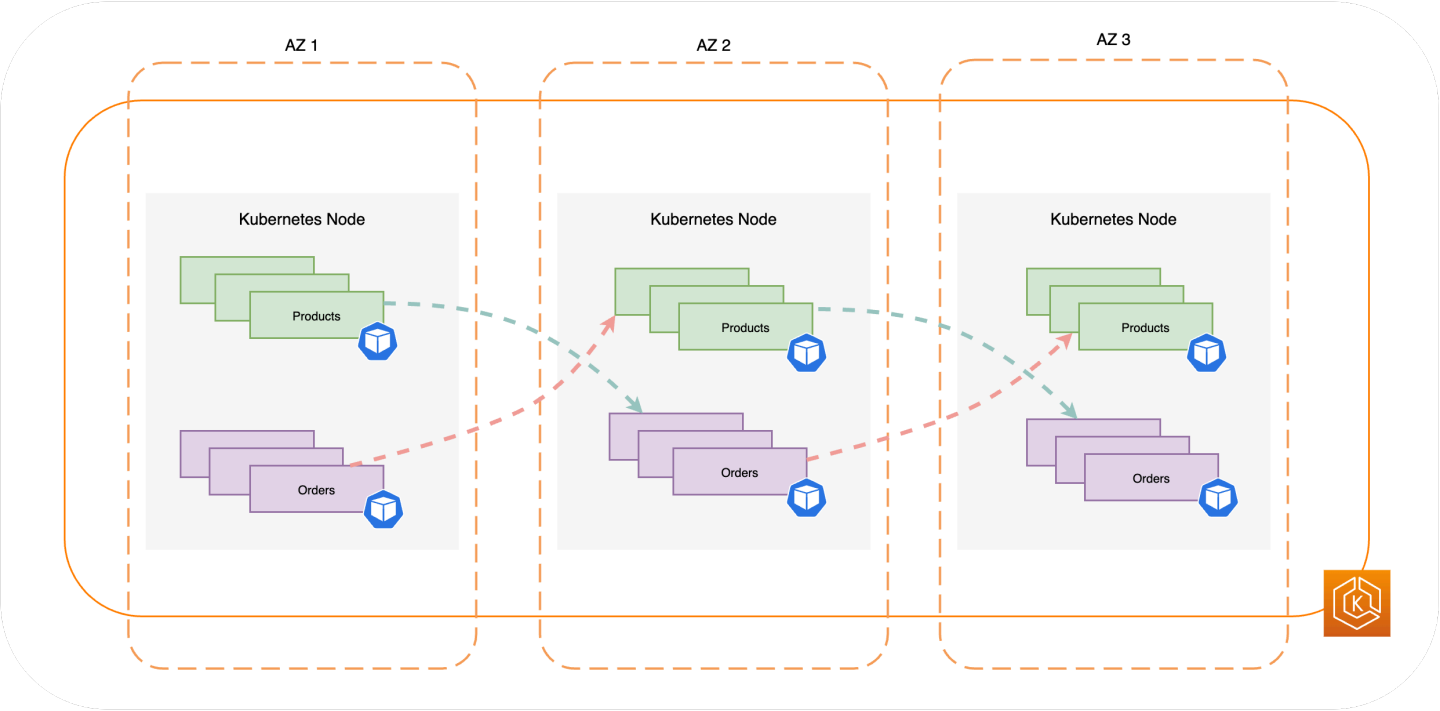
Per ridurre il rischio di questi potenziali impatti negativi, ti consigliamo di eseguire un provisioning eccessivo della capacità di calcolo in alcuni nodi di lavoro di ciascuno degli AZ. In questo modo, il sistema di pianificazione Kubernetes dispone di una capacità pre-esistente disponibile per il posizionamento di nuovi pod, il che è particolarmente importante quando si perde una delle AZ nel proprio ambiente.

## Esegui e distribuisci più repliche di pod tra le zone di disponibilità

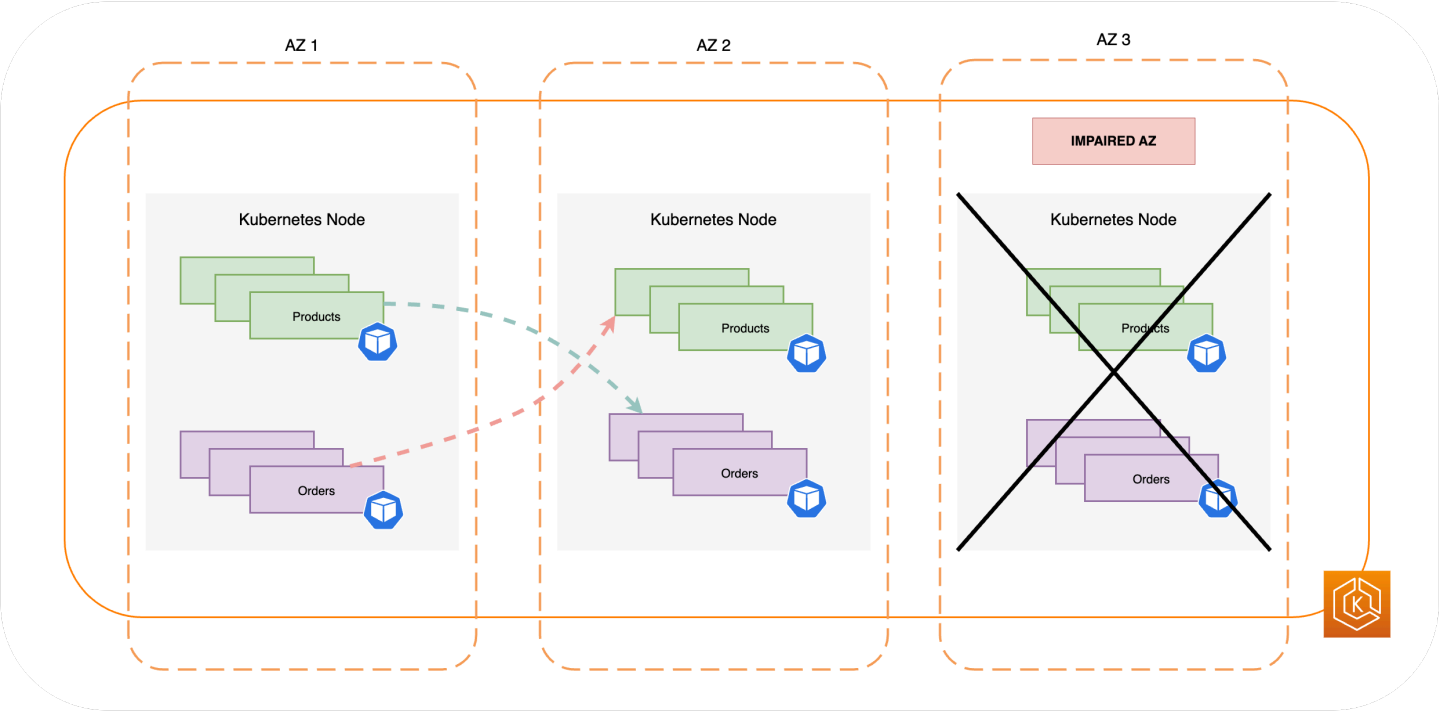
Kubernetes ti consente di pre-scalare i carichi di lavoro eseguendo più istanze (repliche di pod) di una singola applicazione. L'esecuzione di più repliche di pod per un'applicazione elimina i singoli punti di guasto e aumenta le prestazioni complessive riducendo il carico di sovraccarico su una singola replica. Tuttavia, per garantire un'elevata disponibilità e una migliore tolleranza ai guasti per le applicazioni, si consiglia di eseguire più repliche dell'applicazione e di distribuirle su diversi domini

di guasto, noti anche come domini di topologia. I domini di guasto in questo scenario sono le zone di disponibilità. Utilizzando i vincoli di diffusione della topologia, è possibile configurare le applicazioni in modo che abbiano una stabilità statica pre-esistente. Quindi, in caso di una compromissione di AZ, l'ambiente disporrà di un numero sufficiente di repliche in AZ funzionanti per gestire immediatamente eventuali picchi o aumenti improvvisi di traffico.

Il diagramma seguente illustra un ambiente EKS con un flusso di traffico da est a ovest quando tutte le AZ sono integre.




Il diagramma seguente illustra un ambiente EKS con un flusso di traffico da est a ovest in cui una singola AZ ha subito un guasto ed è stato avviato uno spostamento zonale.




Il seguente frammento di codice è un esempio di come configurare il carico di lavoro con più repliche in Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders
spec:
  replicas: 9
  selector:
    matchLabels:
      app: orders
  template:
    metadata:
      labels:
        app: orders
        tier: backend
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: "topology.kubernetes.io/zone"
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchLabels:
```

```
            app: orders
```

Soprattutto, è consigliabile eseguire più repliche del software del server DNS (CoreDNS/kube-dns) e applicare vincoli di diffusione della topologia simili, se non sono configurati per impostazione predefinita. Ciò aiuta a garantire che, in caso di una singola compromissione di AZ, si disponga di un numero sufficiente di pod DNS in AZ integre per continuare a gestire le richieste di rilevamento servizi per altri pod comunicanti nel cluster. Il componente aggiuntivo EKS CoreDNS dispone di impostazioni predefinite per i pod CoreDNS che assicurano che, se sono disponibili nodi in più AZ, siano distribuiti nelle zone di disponibilità del cluster. Se si desidera, è possibile sostituire queste impostazioni predefinite con le proprie configurazioni personalizzate.

Quando installi CoreDNS con Helm, è possibile aggiornare `replicaCount` nel file values.yaml per assicurarti di avere un numero sufficiente di repliche in ciascuna AZ. Inoltre, per garantire che queste repliche siano distribuite tra le diverse AZ nell'ambiente cluster, assicurati di aggiornare la proprietà `topologySpreadConstraints` nello stesso file `values.yaml`. Il seguente frammento di codice illustra come configurare CoreDNS per eseguire tale operazione.

values.yaml CoredNS Helm

```
replicaCount: 6
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

In caso di compromissione di AZ, è possibile assorbire l'aumento del carico sui pod CoreDNS, utilizzando un sistema di scalabilità automatica per CoreDNS. Il numero di istanze DNS necessarie dipende dal numero di carichi di lavoro in esecuzione nel cluster. CoreDNS è legato alla CPU, il che gli consente di scalare in base alla CPU utilizzando l'Horizontal Pod Autoscaler (HPA). Di seguito è riportato un esempio che è possibile modificare in base alle esigenze.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: coredns
  namespace: default
spec:
```

```
  maxReplicas: 20
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coredns
  targetCPUUtilizationPercentage: 50
```

In alternativa, EKS può gestire la scalabilità automatica dell'implementazione CoreDNS nella versione del componente aggiuntivo EKS di CoreDNS. Questo autoscaler di CoreDNS monitora continuamente lo stato del cluster, incluso il numero di nodi e core CPU. Sulla base di tali informazioni, il controller regola dinamicamente il numero di repliche dell'implementazione CoreDNS in un cluster EKS.

Per abilitare la configurazione di scalabilità automatica nel componente aggiuntivo CoredNS EKS, utilizzare la seguente impostazione di configurazione:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

È inoltre possibile utilizzare il NodeLocal DNS o il cluster proportional autoscaler per scalare CoreDNS. Per ulteriori informazioni, consulta Scalare CoredNS orizzontalmente.

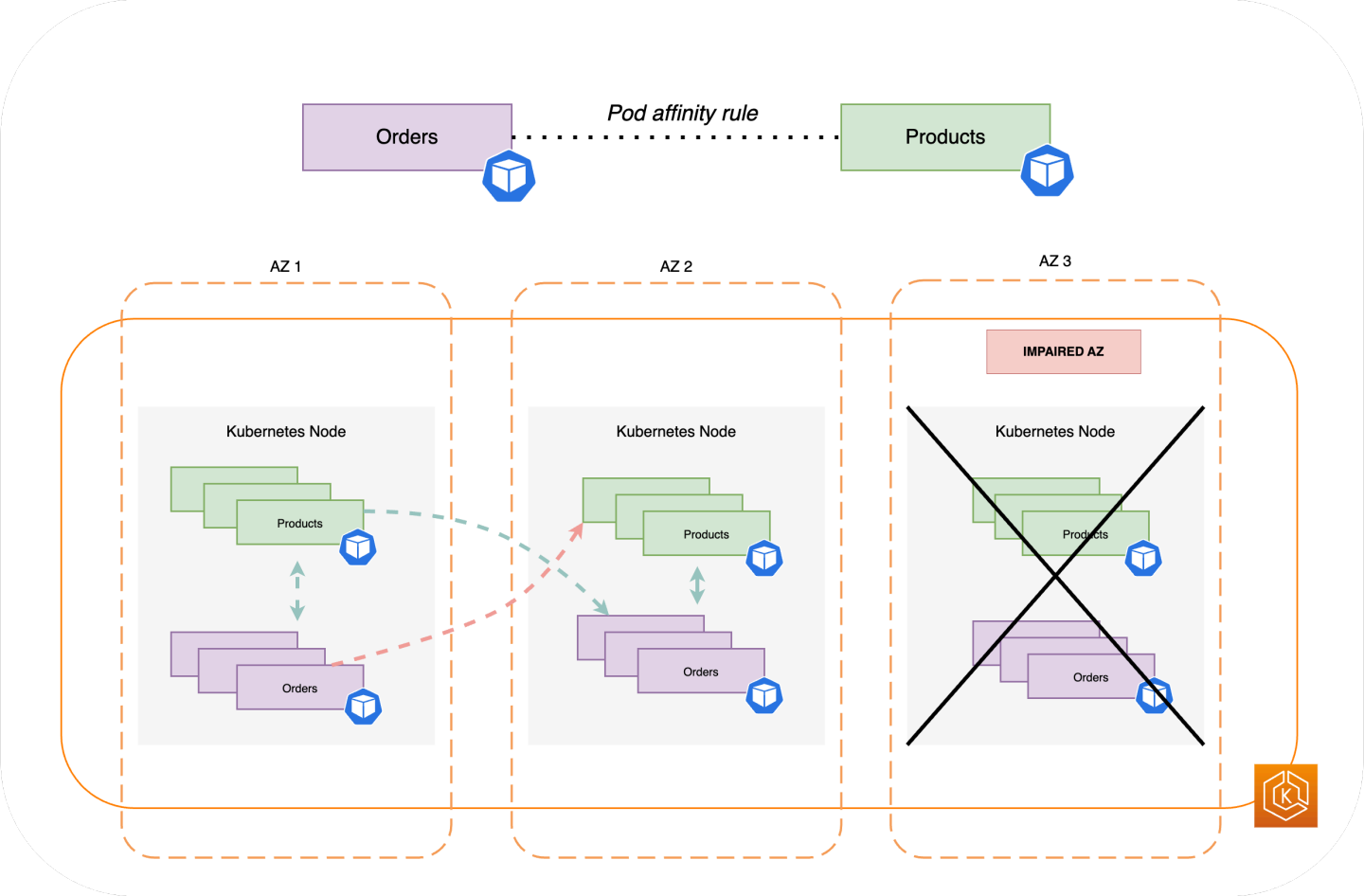
## Co-locare i pod interdipendenti nella stessa zona di disponibilità

In genere, le applicazioni hanno carichi di lavoro distinti che devono comunicare tra loro per completare con successo un processo end-to-end. Se queste applicazioni distinte sono distribuite su diverse AZ e non sono co-locate nella stessa AZ, una singola violazione della AZ può influire sul processo end-to-end. Ad esempio, se l'applicazione A ha più repliche in AZ 1 e AZ 2, ma l'applicazione B ha tutte le repliche in AZ 3, la perdita di AZ 3 influirà sui processi end-to-end tra i due carichi di lavoro, l'applicazione A e l'applicazione B. Se combini i vincoli di diffusione della topologia con l'affinità dei pod, puoi migliorare la resilienza dell'applicazione distribuendo i pod su tutte le AZ. Inoltre, ciò configura una relazione tra determinati pod per garantire che siano co-locati.

Con le regole di affinità dei pod, è possibile definire le relazioni tra i carichi di lavoro per influenzare il comportamento del sistema di pianificazione Kubernetes in modo che collochi i pod sullo stesso nodo worker o nella stessa AZ. Inoltre, è possibile configurare la rigidità dei vincoli di pianificazione.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: products
  namespace: ecommerce
  labels:
    app.kubernetes.io/version: "0.1.6"
spec:
  template:
    spec:
      serviceAccountName: graphql-service-account
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - orders
            topologyKey: "kubernetes.io/hostname"
```

Il diagramma seguente mostra diversi pod che sono stati co-locati sullo stesso nodo utilizzando le regole di affinità dei pod.




## Verifica che il tuo ambiente cluster sia in grado di gestire la perdita di una AZ

Dopo aver completato i requisiti descritti nelle sezioni precedenti, il passaggio successivo consiste nel verificare di disporre di una capacità di elaborazione e carico di lavoro sufficiente per gestire la perdita di una AZ. Per eseguire tale operazione è possibile avviare manualmente uno spostamento zonale in EKS. In alternativa, è possibile abilitare lo spostamento zonale automatico e configurare le sessioni pratiche, che consentono anche di verificare che le applicazioni funzionino come previsto con una AZ in meno nell'ambiente cluster.

# Domande frequenti

### Qual è il vantaggio di utilizzare questa funzionalità?

Utilizzando lo spostamento zonale ARC o lo spostamento zonale automatico nel cluster EKS, è possibile mantenere meglio la disponibilità delle applicazioni Kubernetes automatizzando il processo di ripristino rapido che consiste nello spostamento del traffico di rete interno al cluster da una zona AZ

compromessa. Con ARC, è possibile evitare passaggi lunghi e complicati, che possono portare a un periodo di recupero prolungato in caso di eventi AZ compromessi.

Come funziona questa funzionalità con altri servizi? AWS

EKS si integra con ARC, che fornisce l'interfaccia principale in cui eseguire le operazioni di ripristino. AWS Per garantire che il traffico all'interno del cluster sia indirizzato in modo appropriato lontano da una zona di disponibilità compromessa, EKS apporta modifiche all'elenco degli endpoint di rete per i pod in esecuzione sul piano dati Kubernetes. Se utilizzi Elastic Load Balancing per instradare il traffico esterno verso il cluster, è possibile registrare i bilanciatori del carico con ARC e avviare uno spostamento zonale su di essi per impedire che il traffico fluisca verso la zona AZ deteriorata. Se si utilizza la modalità EKS Auto, la modalità EKS Auto limita automaticamente il provisioning dei nodi agli AZ integri. Lo spostamento zonale funziona anche con i gruppi di Amazon EC2 Auto Scaling creati da gruppi di nodi gestiti da EKS. Per evitare che una AZ compromessa sia utilizzata per nuovi pod Kubernetes o avvii di nodi, EKS rimuove la AZ compromessa dai gruppi Auto Scaling.

In che modo questa funzionalità è diversa dalle protezioni Kubernetes predefinite?

Tale funzionalità funziona in combinazione con diverse protezioni integrate di Kubernetes che aiutano la resilienza delle applicazioni dei clienti. È possibile configurare le sonde readiness e liveness di pod che decidono quando un Pod deve ricevere traffico. Quando queste sonde si guastano, Kubernetes rimuove questi pod come obiettivi di un servizio e il traffico non è più inviato al pod. Sebbene ciò sia utile, non è semplice per i clienti configurare tali controlli dell'integrità in modo da garantire che abbiano esito negativo in caso di deterioramento di una AZ. La funzione di spostamento zonale ARC fornisce una rete di sicurezza aggiuntiva che ti aiuta a isolare completamente una AZ deteriorata quando le protezioni native di Kubernetes non erano sufficienti. Lo spostamento zonale offre anche un modo semplice per testare la prontezza operativa e la resilienza dell'architettura.

Posso AWS iniziare un passaggio zonale per mio conto?

Sì, se desideri un modo completamente automatizzato di utilizzare lo spostamento zonale ARC, puoi abilitare lo spostamento zonale automatico ARC. Con lo spostamento automatico zonale, puoi contare sul AWS monitoraggio dello stato degli AZ per il tuo cluster EKS e sull'avvio automatico di uno spostamento zonale quando viene rilevata una compromissione dell'AZ.

Cosa succede se utilizzo questa funzionalità e i miei nodi worker e i miei carichi di lavoro non sono predimensionati?

Se non si esegue una pre-scalabilità e si affida al provisioning di nodi o pod aggiuntivi durante uno spostamento zonale, si rischia un ripristino ritardato. Il processo di aggiunta di nuovi nodi al piano

dati di Kubernetes richiede del tempo, il che può influire sulle prestazioni e sulla disponibilità in tempo reale delle applicazioni, soprattutto in caso di compromissione zonale. Inoltre, in caso di compromissione zonale, potresti riscontrare un potenziale limite di capacità di elaborazione che potrebbe impedire l'aggiunta dei nuovi nodi necessari alle AZ integre.

Se utilizzi la modalità EKS Auto, EKS Auto Mode esegue automaticamente il provisioning di nuovi nodi in AzS integri per soddisfare la domanda non programmata di Pod. Tuttavia, i nuovi nodi richiedono ancora tempo per essere avviati e pronti. Per un ripristino più rapido, consigliamo di prescalare i carichi di lavoro su più AZ.

Se i carichi di lavoro non sono pre-scalati e distribuiti su tutte le AZ del cluster, una compromissione zonale potrebbe influire sulla disponibilità di un'applicazione che è eseguita solo sui nodi worker nelle AZ interessate. Per mitigare il rischio di un'interruzione completa della disponibilità dell'applicazione, EKS dispone di un sistema a prova di guasto per l'invio del traffico agli endpoint di pod in una zona compromessa se il carico di lavoro ha tutti i suoi endpoint in una AZ non integra. Tuttavia, consigliamo vivamente di pre-scalare e distribuire le applicazioni su tutte le AZ per mantenere la disponibilità in caso di problemi zonali.

Come funziona se sto eseguendo un'applicazione stateful?

Se stai eseguendo un'applicazione stateful, devi valutarne la tolleranza ai guasti, in base al tuo caso d'uso e all'architettura. Se hai un' active/standby architettura o un pattern, potrebbero esserci casi in cui l'attivo si trova in un AZ compromesso. A livello di applicazione, se lo standby non è attivato, è possibile che si verifichino problemi con l'applicazione. Inoltre, potresti riscontrare problemi quando sono avviati nuovi pod Kubernetes in AZ integre, poiché non saranno in grado di collegarsi ai volumi persistenti limitati alla AZ compromessa.

Questa funzionalità funziona con EKS Auto Mode?

Sì. Quando si abilita lo spostamento zonale su un cluster EKS Auto Mode, EKS Auto Mode risponde automaticamente agli eventi di spostamento zonale. Durante uno spostamento zonale, EKS Auto Mode interrompe il provisioning di nuovi nodi nell'AZ compromesso, sospende le azioni volontarie di interruzione (come consolidamento e deriva) che potrebbero influire sull'AZ compromesso ed evita di aumentare la capacità di lancio di Pod con rigidi requisiti di pianificazione destinati all'AZ compromesso. Non è necessaria alcuna configurazione aggiuntiva oltre all'abilitazione dello spostamento zonale sul cluster.

Questa funzionalità funziona con Karpenter autogestito?

Self-managed Il supporto Karpenter è disponibile con lo spostamento zonale ARC e lo spostamento automatico zonale in EKS con Karpenter versione 1.12 o successiva. https://github.com/aws/karpenter-provider-aws

Questa funzionalità funziona con EKS Fargate?

Questa funzionalità non funziona con EKS Fargate. Per impostazione predefinita, quando EKS Fargate riconosce un evento di integrità zonale, i pod preferiranno essere eseguiti nelle altre AZ.

Il piano di controllo Kubernetes gestito da EKS ne risentirà?

No, per impostazione predefinita Amazon EKS esegue ed effettua il dimensionamento di istanze del piano di controllo Kubernetes in molteplici AZ per garantire un'elevata disponibilità. Lo spostamento zonale ARC e lo spostamento zonale automatico agiscono solo sul piano dati Kubernetes.

Ci sono dei costi associati a questa nuova funzionalità?

È possibile utilizzare uno spostamento zonale ARC e uno spostamento zonale automatico nel cluster EKS senza costi aggiuntivi. Tuttavia, continuerai a pagare per le istanze assegnate e consigliamo vivamente di pre-scalare il tuo piano dati Kubernetes prima di utilizzare questa funzionalità. È necessario considerare un equilibrio tra costi e disponibilità delle applicazioni.

### Risorse aggiuntive

- Come funziona uno spostamento zonale
- Best practice per spostamenti zonali in ARC
- Risorse e scenari supportati per lo spostamento zonale e lo spostamento zonale automatico
- Funzionamento di carichi di lavoro resilienti su Amazon EKS
- Elimina il ritardo di scalabilità dei nodi Kubernetes con la priorità dei pod e l'eccesso di provisioning
- Scala i pod CoreDNS per un traffico DNS elevato

## Abilitazione dello spostamento zonale EKS per evitare zone di disponibilità compromesse

Amazon Application Recovery Controller (ARC) aiuta a gestire e coordinare il ripristino per le tue applicazioni nelle zone di disponibilità (AZ) e funziona con molti servizi, tra cui Amazon EKS. Con il supporto EKS per lo spostamento zonale ARC, è possibile spostare il traffico di rete interno al cluster

da una AZ compromessa. Puoi anche autorizzare AWS a monitorare lo stato delle tue AZ e spostare temporaneamente il traffico di rete da una zona non sana per tuo conto.

Come utilizzare uno spostamento zonale EKS:

1. Abilita il tuo cluster EKS con Amazon Application Recovery Controller (ARC). Questa operazione viene eseguita a livello di cluster utilizzando la console Amazon EKS, la AWS CLI o CloudFormation eksctl.
2. Una volta abilitato, puoi gestire i turni zonali o gli spostamenti automatici zonali utilizzando la console ARC, la AWS CLI o le API zonal shift e zonal autoshift.

Tieni presente che dopo aver registrato un cluster EKS con ARC, devi comunque configurare ARC. Per esempio, è possibile utilizzare la console ARC per configurare lo spostamento zonale automatico.

Per informazioni più dettagliate su come funziona lo spostamento zonale EKS e su come progettare i carichi di lavoro per gestire le zone di disponibilità compromesse, consulta the section called "Ulteriori informazioni relative allo spostamento zonale".

## Considerazioni

- Consigliamo di attendere almeno 60 secondi tra le operazioni di spostamento zonale per garantire la corretta elaborazione di ciascuna richiesta.

  Quando si tenta di eseguire turni zonali in rapida successione (entro 60 secondi l'uno dall'altro), il servizio Amazon EKS potrebbe non elaborare correttamente tutte le richieste di turni. Ciò è dovuto all'attuale meccanismo di polling che aggiorna lo stato zonale del cluster. Se è necessario eseguire più spostamenti zonali, assicuratevi che vi sia un periodo di tempo adeguato tra le operazioni affinché il sistema elabori ogni modifica.

## What is Amazon Application Recovery Controller?

Amazon Application Recovery Controller (ARC) ti aiuta a prepararti e a portare a termine ripristini più rapidi per le applicazioni in esecuzione su AWS. Lo spostamento di zona consente di ripristinare rapidamente i problemi relativi alla zona di disponibilità (AZ), spostando temporaneamente il traffico di una risorsa supportata da una zona di disponibilità a favore di aree di disponibilità efficienti nella regione. AWS

Ulteriori informazioni su Amazon Application Recovery Controller (ARC)

## Che cos'è uno spostamento zonale?

Lo spostamento zonale è una funzionalità di ARC che consente di spostare il traffico per una risorsa come un cluster EKS o un Elastic Load Balancer lontano da una zona di disponibilità in AWS una regione per mitigare rapidamente un problema e ripristinare rapidamente l'applicazione. È possibile scegliere di spostare il traffico, per esempio, perché una cattiva implementazione causa problemi di latenza o perché la zona di disponibilità è compromessa. Lo spostamento zonale non richiede passaggi di configurazione avanzati. Tuttavia, i carichi di lavoro devono soddisfare determinati requisiti per essere resilienti a livello di zona prima di iniziare un turno. Per ulteriori informazioni, consulta the section called "Ulteriori informazioni relative allo spostamento zonale".

Ulteriori informazioni sullo spostamento zonale ARC

## Cos'è lo spostamento automatico zonale automatico?

L'autoshift zonale è una funzionalità di ARC che puoi abilitare per autorizzare a spostare il traffico da una zona di zona AWS alle risorse supportate, per tuo conto, verso AZ funzionanti nella regione. AWS AWS avvia uno spostamento automatico quando la telemetria interna indica che in una zona AZ in una regione si è verificata una compromissione che potrebbe avere un impatto sui clienti. La telemetria interna incorpora metriche provenienti da più fonti, tra cui la AWS rete e i servizi Amazon EC2 ed Elastic Load Balancing.

AWS interrompe gli spostamenti automatici quando gli indicatori indicano che non esiste più un problema o un problema potenziale.

Ulteriori informazioni sullo spostamento zonale automatico ARC

## Cosa fa EKS durante un cambio di zona?

EKS aggiorna le configurazioni di rete per evitare di indirizzare il traffico verso AZ compromesse. Il comportamento di calcolo specifico dipende dalla configurazione del piano dati del cluster:

- EKS Auto Mode: EKS Auto Mode interrompe automaticamente il provisioning di nuovi nodi nella zona AZ compromessa, interrompe le interruzioni volontarie dei nodi (come il consolidamento) che potrebbero influire sulla zona AZ compromessa ed evita di avviare la capacità di avvio per Pod con requisiti di pianificazione rigorosi rivolti alla zona AZ compromessa (ad esempio associazioni di volume persistenti o vincoli rigorosi di diffusione della topologia). Non è necessaria alcuna configurazione aggiuntiva oltre all'abilitazione dello spostamento zonale sul cluster.

- Self-managed Karpenter (v1.12+): se utilizzi Karpenter v1.12 o versione successiva con il `--enable-zonal-shift` flag, Karpenter interrompe il provisioning della capacità nella zona AZ compromessa e interrompe le interruzioni volontarie che dipendono dalla zona AZ compromessa.
- Gruppi di nodi gestiti: EKS lancerà nuovi nodi nelle AZ integre solo durante un cambio di zona sospendendo il ribilanciamento delle zone di disponibilità e aggiornando il gruppo Auto Scaling.

Quando il turno scade o viene annullato, le configurazioni di rete e il comportamento di calcolo vengono ripristinati per includere l'AZ che in precedenza era stato rilevato come non integro.

Ulteriori informazioni sullo spostamento zonale EKS

## Registra il cluster EKS con Amazon Application Recovery Controller (ARC) (AWS console)

1. Trova il nome e la regione del cluster EKS che desideri registrare con ARC.
2. Vai alla Console EKS in quella regione e seleziona il tuo cluster.
3. Nella pagina delle Informazioni sul cluster, seleziona la scheda Panoramica.
4. Sotto l'intestazione Spostamento zonale, seleziona il pulsante Gestisci.
5. Seleziona abilita o disabilita per lo spostamento zonale EKS.

Ora il tuo cluster EKS è registrato con ARC.

Se si desidera AWS rilevare ed evitare zone di disponibilità ridotta, è necessario configurare l'autoshift zonale ARC. Ad esempio, è possibile eseguire questa operazione nella console ARC.

## Fasi successive

- Scopri come abilitare lo spostamento zonale automatico.
- Scopri come avviare manualmente uno spostamento zonale.

# Informazioni su come funziona il controllo degli accessi in Amazon EKS

Scopri come gestire l'accesso al tuo cluster Amazon EKS. L'utilizzo di Amazon EKS richiede la conoscenza di come Kubernetes e AWS Identity and Access Management (AWS IAM) gestiscono il controllo degli accessi.

Questa sezione include:

the section called "Accesso API Kubernetes": scopri come permettere alle applicazioni o agli utenti di autenticarsi nell'API Kubernetes. Puoi utilizzare access entry, aws-auth ConfigMap o un provider OIDC esterno.

the section called "Accesso alle risorse di cluster"— Scopri come configurare la comunicazione con Console di gestione AWS il tuo cluster Amazon EKS. Utilizza la console per visualizzare le risorse Kubernetes nel cluster, come namespace, nodi e pod.

the section called "Modifica le risorse del cluster"— Scopri le autorizzazioni necessarie per modificare le risorse Kubernetes.

the section called "Accedere al cluster con kubectl" : scopri come configurare kubectl per comunicare con il tuo cluster Amazon EKS. Usa la AWS CLI per creare un file kubeconfig.

the section called "Accesso al carico di lavoro a AWS "— Scopri come associare un account di servizio Kubernetes a IAM Roles. AWS Puoi usare Pod Identity o i ruoli IAM per gli account di servizio (IRSA).

## Attività comuni

- Concedi l'accesso all'API Kubernetes agli sviluppatori. Visualizza le risorse Kubernetes in. Console di gestione AWS
  - Soluzione: utilizza le voci di accesso per associare le autorizzazioni RBAC di Kubernetes agli utenti o ai ruoli IAM. AWS
- Configura kubectl per comunicare con un cluster Amazon EKS utilizzando le credenziali. AWS
  - Soluzione: utilizzate la AWS CLI per creare un file kubeconfig.
- Usa un provider di identità esterno, come Ping Identity, per autenticare gli utenti nell'API Kubernetes.

  - Soluzione: collega un provider OIDC esterno.
- Concedi ai carichi di lavoro sul tuo cluster Kubernetes la possibilità di chiamare. AWS APIs
  - Soluzione: utilizza Pod Identity per associare un ruolo AWS IAM a un account di servizio Kubernetes.

## Contesto

- Scopri come funzionano gli account di servizio Kubernetes.
- Esamina il modello di controllo degli accessi basato sui ruoli (Role Based Access Control, RBAC) di Kubernetes
- Per ulteriori informazioni sulla gestione dell'accesso alle AWS risorse, consulta la AWS IAM User Guide. In alternativa, segui un corso introduttivo gratuito sull'uso AWS di IAM.

## Considerazioni relative alla modalità automatica di EKS

EKS Auto Mode si integra con EKS Pod Identity e le voci di accesso EKS.

- La modalità automatica di EKS usa le voci di accesso per concedere le autorizzazioni Kubernetes del piano di controllo di EKS. Ad esempio, le policy di accesso consentono alla modalità automatica di EKS di leggere informazioni sugli endpoint e sui servizi di rete.
  - Non puoi disabilitare le voci di accesso su un cluster della modalità automatica di EKS.
  - Facoltativamente, puoi abilitare `aws-auth ConfigMap`.
  - Le voci di accesso per la modalità automatica di EKS sono configurate automaticamente. Puoi visualizzare queste voci di accesso, ma non puoi modificarle.
  - Se utilizzi un NodeClass ruolo Node IAM personalizzato, devi creare una voce di accesso per il ruolo utilizzando la policy di EKSAuto NodePolicy accesso Amazon.
- Se desideri concedere ai carichi di lavoro le autorizzazioni per AWS i servizi, usa EKS Pod Identity.
  - Non è necessario installare l'agente Pod Identity sui cluster della modalità automatica di EKS.

## Concedere agli utenti e ai ruoli IAM l'accesso alle API Kubernetes

Il tuo cluster dispone di un endpoint dell'API Kubernetes. Kubectl utilizza questa API. Puoi autenticarti su questa API usando due tipi di identità:

- Un principale (ruolo o utente) di AWS Identity and Access Management (IAM): questo tipo richiede l'autenticazione su IAM. Gli utenti possono accedere AWS come utenti IAM o con un'identità federata utilizzando le credenziali fornite tramite una fonte di identità. Gli utenti possono accedere solo con un'identità federata se l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando gli utenti accedono AWS utilizzando la federazione, assumono indirettamente un ruolo. Quando gli utenti utilizzano questo tipo di identità:
  - Puoi assegnare loro autorizzazioni Kubernetes in modo che possano lavorare con gli oggetti Kubernetes sul tuo cluster. Per ulteriori informazioni su come assegnare le autorizzazioni ai principali IAM in modo che possano accedere agli oggetti Kubernetes sul cluster, consulta la sezione the section called "Voci di accesso".
  - Puoi assegnare loro le autorizzazioni IAM in modo che possano lavorare con il tuo cluster Amazon EKS e le sue risorse utilizzando l'API Amazon EKS, la AWS CLI,, AWS CloudFormation o. Console di gestione AWS`eksctl` Per ulteriori informazioni, consulta Actions defined by Amazon Elastic Kubernetes Service nella Documentazione di riferimento per l'autorizzazione ai servizi.
  - I nodi si aggiungono al cluster assumendo un ruolo IAM. La capacità di accedere al cluster utilizzando i principali (IAM) è abilitata dall'autenticatore AWS IAM per Kubernetes, che viene eseguito sul piano di controllo di Amazon EKS.
- Un utente del tuo provider OpenID Connect (OIDC): questo tipo richiede l'autenticazione al tuo provider OIDC. Per ulteriori informazioni sulla configurazione del provider OIDC con il cluster Amazon EKS, consulta the section called "Collegare il provider OIDC". Quando gli utenti utilizzano questo tipo di identità:
  - Puoi assegnare loro autorizzazioni Kubernetes in modo che possano lavorare con gli oggetti Kubernetes sul tuo cluster.
  - Non puoi assegnare loro autorizzazioni IAM in modo che possano lavorare con il tuo cluster Amazon EKS e le sue risorse utilizzando l'API Amazon EKS, la AWS CLI,, AWS CloudFormation o. Console di gestione AWS`eksctl`

Puoi utilizzare entrambi i tipi di identità con il tuo cluster. Il metodo di autenticazione IAM non può essere disabilitato. Il metodo di autenticazione OIDC è facoltativo.

## Associa le identità IAM alle autorizzazioni Kubernetes

L'Autenticatore AWS IAM per Kubernetes è installato sul piano di controllo del tuo cluster. Abilita i principali AWS Identity and Access Management (ruoli e utenti) a cui consenti di accedere alle risorse

Kubernetes sul tuo cluster. Puoi consentire ai principali IAM di accedere agli oggetti Kubernetes sul tuo cluster utilizzando uno dei metodi seguenti:

- Creazione di voci di accesso: se la versione del cluster corrisponde o è successiva alla versione della piattaforma elencata nella sezione Prerequisiti per la versione del cluster, ti consigliamo di utilizzare questa opzione.

  Utilizza le voci di accesso per gestire le autorizzazioni Kubernetes dei principali IAM dall'esterno del cluster. Puoi aggiungere e gestire l'accesso al cluster utilizzando l'API EKS, l'interfaccia a riga di AWS comando, AWS gli SDK e. AWS CloudFormation Console di gestione AWS Ciò significa che puoi gestire gli utenti con gli stessi strumenti con cui hai creato il cluster.

  Per iniziare, segui Cambia la modalità di autenticazione per utilizzare le voci di accesso, quindi Migrazione delle voci aws-auth ConfigMap esistenti alle voci di accesso.
- Aggiunta di voci a **`aws-auth ConfigMap`**: se la versione della piattaforma del cluster è precedente alla versione elencata nella sezione Prerequisiti, è necessario utilizzare questa opzione. Se la versione della piattaforma del cluster corrisponde o è successiva a quella elencata nella sezione Prerequisiti per la versione di Kubernetes del cluster e hai aggiunto delle voci a `ConfigMap`, ti consigliamo di eseguire la migrazione di tali voci per potervi accedere. Tuttavia, non puoi migrare le voci aggiunte da Amazon EKS a `ConfigMap`, come le voci per i ruoli IAM utilizzati con gruppi di nodi gestiti o profili Fargate. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes".
  - Se è necessario utilizzare l'opzione `aws-auth ConfigMap`, è possibile aggiungere voci a `ConfigMap` utilizzando il comando `eksctl create iamidentitymapping`. Per ulteriori informazioni, consulta Manage IAM users and roles nella documentazione di `eksctl`.

## Set Cluster Authentication Mode

Ogni cluster dispone di una modalità di autenticazione. La modalità di autenticazione determina quali metodi è possibile utilizzare per consentire ai principali IAM di accedere agli oggetti Kubernetes sul cluster. Sono disponibili tre modalità di autenticazione.

> **Important**
>
> Una volta abilitato, il metodo di immissione dell'accesso non può essere disabilitato.

Se il metodo `ConfigMap` non è abilitato durante la creazione del cluster, non può essere abilitato in seguito. Tutti i cluster creati prima dell'introduzione delle voci di accesso hanno il metodo `ConfigMap` abilitato.
Se si utilizzano nodi ibridi con il cluster, è necessario utilizzare le modalità di autenticazione del cluster `API` o `API_AND_CONFIG_MAP`.

Il **`aws-auth` `ConfigMap`** all'interno del cluster

Questa è la modalità di autenticazione originale per i cluster Amazon EKS. Il principale IAM che ha creato il cluster è l'utente iniziale che può accedere al cluster utilizzando `kubectl`. L'utente iniziale deve aggiungere altri utenti all'elenco in `aws-auth` `ConfigMap` e assegnare autorizzazioni che influiscono sugli altri utenti all'interno del cluster. Questi altri utenti non possono gestire o rimuovere l'utente iniziale, poiché non c'è una voce da gestire in `ConfigMap`.

Sia le voci di accesso che quelle di accesso **`ConfigMap`**

Con questa modalità di autenticazione, è possibile utilizzare entrambi i metodi per aggiungere i principali IAM al cluster. Si noti che ogni metodo memorizza voci separate; ad esempio, se si aggiunge una voce di accesso dalla AWS CLI, non `aws-auth` `ConfigMap` viene aggiornata.

Accedi solo alle voci

Con questa modalità di autenticazione, puoi utilizzare l'API EKS, l'interfaccia a riga di AWS comando AWS CloudFormation, AWS gli SDK e Console di gestione AWS gestire l'accesso al cluster per i principali IAM.

Ogni voce di accesso include un tipo, che consente di utilizzare la combinazione di un ambito di accesso per limitare il principale a un namespace specifico e una policy di accesso per definire policy di autorizzazione riutilizzabili preconfigurate. In alternativa, è possibile utilizzare il tipo STANDARD e i gruppi Kubernetes RBAC per assegnare autorizzazioni personalizzate.

| Modalità di autenticazione | Metodi |
| --- | --- |
| Solo `ConfigMap` (`CONFIG_MAP`) | `aws-auth` `ConfigMap` |
| API EKS e `ConfigMap` (`API_AND_CONFIG_MAP`) | accedi alle voci nell'API EKS, nell'interfaccia AWS a riga di comando, negli AWS SDK e |

| Modalità di autenticazione | Metodi |
| --- | --- |
| | AWS CloudFormation Console di gestione AWS `aws-auth ConfigMap` |
| Solo API EKS (`API`) | accedere alle voci nell'API EKS, nell'interfaccia a riga di AWS comando, negli AWS SDK e AWS CloudFormation Console di gestione AWS |

> **Note**
>
> La modalità automatica di Amazon EKS richiede le voci di accesso.

# Concedere agli utenti IAM l'accesso a Kubernetes con le voci di accesso EKS

Questa sezione è progettata per mostrarti come gestire l'accesso principale IAM ai cluster di Kubernetes in Amazon Elastic Kubernetes Service (EKS) utilizzando voci di accesso e policy. Sono disponibili dettagli sulla modifica delle modalità di autenticazione, la migrazione da voci `aws-auth` ConfigMap precedenti, la creazione, l'aggiornamento e l'eliminazione delle voci di accesso, l'associazione delle policy alle voci, la revisione delle autorizzazioni delle policy predefinite e i prerequisiti e le considerazioni chiave per la gestione sicura degli accessi.

## Panoramica

Le voci di accesso EKS sono il modo migliore per concedere agli utenti l'accesso all'API Kubernetes. Ad esempio, puoi utilizzare le voci di accesso per concedere agli sviluppatori l'accesso all'uso di kubectl. Fondamentalmente, una voce di accesso EKS associa un set di autorizzazioni di Kubernetes a un'identità IAM, ad esempio un ruolo IAM. Ad esempio, uno sviluppatore può assumere un ruolo IAM e utilizzarlo per autenticarsi in un cluster EKS.

## Funzionalità

- Autenticazione e autorizzazione centralizzate: controlla l'accesso ai cluster di Kubernetes direttamente tramite le API di Amazon EKS, eliminando la necessità di passare fra AWS e le API Kubernetes per le autorizzazioni utente.

- Gestione granulare delle autorizzazioni: utilizza le voci e le policy di accesso per definire autorizzazioni granulari per i principali IAM AWS, inclusa la modifica o la revoca dell'accesso da amministratore del cluster dal creatore.
- Integrazione con strumenti IaC: supporta l'infrastruttura sotto forma di strumenti di codice come AWS CloudFormation, Terraform e AWS CDK per definire le configurazioni di accesso durante la creazione del cluster.
- Ripristino della configurazione errata: consente di ripristinare l'accesso al cluster tramite l'API Amazon EKS senza accesso diretto all'API Kubernetes.
- Sovraccarico ridotto e sicurezza avanzata: centralizza le operazioni per ridurre il sovraccarico sfruttando al contempo funzionalità IAM AWS come la registrazione degli audit CloudTrail e l'autenticazione a più fattori.

## Come allegare le autorizzazioni

Puoi allegare le autorizzazioni Kubernetes per accedere alle voci in due modi:

- Utilizzare una policy di accesso. Le policy di accesso sono modelli di autorizzazioni di Kubernetes predefiniti gestiti da AWS. Per ulteriori informazioni, consulta the section called "Rivedere le policy di accesso".
- Fare riferimento a un gruppo di Kubernetes. Se associ un'identità IAM a un gruppo di Kubernetes, puoi creare risorse di Kubernetes che concedono le autorizzazioni al gruppo. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

## Considerazioni

Quando si abilitano le voci di accesso EKS sui cluster esistenti, tenere presente quanto segue:

- Comportamento legacy dei cluster: per i cluster creati prima dell'introduzione delle voci di accesso (quelli con versioni iniziali della piattaforma precedenti a quelle specificate in Platform version requirements), EKS crea automaticamente una voce di accesso che riflette le autorizzazioni preesistenti. Questa voce include l'identità IAM che ha originariamente creato il cluster e le autorizzazioni amministrative concesse a tale identità durante la creazione del cluster.
- Gestione di **`aws-auth`** ConfigMap legacy: se il cluster si basa su `aws-auth` ConfigMap legacy per la gestione degli accessi, dopo aver abilitato le voci di accesso, viene creata automaticamente solo la voce di accesso per il creatore del cluster originale. I ruoli o le autorizzazioni aggiuntivi aggiunti a ConfigMap (ad esempio, ruoli IAM personalizzati per sviluppatori o servizi) non vengono

migrati automaticamente. Per risolvere questo problema, creare manualmente le voci di accesso corrispondenti.

## Inizia a usare

1. Determina l'identità IAM e la policy di accesso che desideri utilizzare.
   - the section called "Rivedere le policy di accesso"
2. Abilita le voci di accesso EKS sul cluster. Conferma di avere una versione della piattaforma supportata.
   - the section called "Modalità di autenticazione"
3. Creare una voce di accesso che associ un'identità IAM all'autorizzazione di Kubernetes.
   - the section called "Creare voci di accesso"
4. Effettuare l'autenticazione nel cluster utilizzando l'identità IAM.
   - the section called "Configurare la AWS CLI"
   - the section called "Configura kubectl ed eksctl"

## Associare le policy di accesso alle voci di accesso

È possibile assegnare una o più policy di accesso alle voci di accesso di tipo `STANDARD`. Amazon EKS concede automaticamente agli altri tipi di voci di accesso le autorizzazioni necessarie per funzionare correttamente nel cluster. Le policy di accesso di Amazon EKS includono le autorizzazioni di Kubernetes, non le autorizzazioni IAM. Prima di associare una policy di accesso a una voce di accesso, assicurati di conoscere le autorizzazioni di Kubernetes incluse in ogni policy di accesso. Per ulteriori informazioni, consulta the section called "Rivedere le policy di accesso". Se nessuna delle policy di accesso soddisfa i requisiti, non associarne alcuna a una voce di accesso. Specificare invece uno o più nomi dei gruppi per la voce di accesso, e creare e gestire oggetti di controllo degli accessi basato sui ruoli di Kubernetes. Per ulteriori informazioni, consulta the section called "Creare voci di accesso".

- Una voce di accesso esistente. Per crearne uno, consulta the section called "Creare voci di accesso".
- Un ruolo o utente di gestione delle AWS identità e degli accessi con le seguenti autorizzazioni: `ListAccessEntries`,`DescribeAccessEntry`,`UpdateAccessEntry`,, `ListAccessPolicies`,`AssociateAccessPolicy`, `eDisassociateAccessPolicy`. Per

ulteriori informazioni, consulta Actions defined by Amazon Elastic Kubernetes Service in Service Authorization Reference.

Prima di associare policy di accesso a voci di accesso, considera i seguenti requisiti:

- È possibile associare più policy di accesso a ciascuna voce di accesso, ma è possibile associare ogni policy a una voce di accesso solo una volta. Se associ più policy di accesso, il principale IAM della voce di accesso dispone di tutte le autorizzazioni incluse in tutte le policy di accesso associate.
- Puoi applicare una policy di accesso a tutte le risorse di un cluster o specificando il nome di uno o più namespace di Kubernetes. Puoi utilizzare caratteri jolly per il nome di un namespace. Ad esempio, se desideri applicare una policy di accesso a tutti i namespace che iniziano con `dev-`, puoi specificare `dev-*` come nome del namespace. Assicurati che i namespace siano presenti sul cluster e che quanto digitato corrisponda esattamente al nome reale del namespace sul cluster. Amazon EKS, infatti, non conferma l'ortografia o l'esistenza dei namespace sul cluster.
- È possibile modificare l'ambito di accesso per una policy di accesso dopo averla associata a una voce di accesso. Se hai limitato l'ambito della policy di accesso ai namespace di Kubernetes, puoi aggiungere e rimuovere i namespace per l'associazione, secondo necessità.
- Se associ una policy di accesso a una voce di accesso in cui sono specificati anche i nomi dei gruppi, il principale IAM dispone di tutte le autorizzazioni in tutte le policy di accesso associate. Inoltre, dispone di tutte le autorizzazioni in qualsiasi oggetto `Role` o `ClusterRole` di Kubernetes specificato in qualsiasi oggetto `Role` e `RoleBinding` di Kubernetes che specifica i nomi dei gruppi.
- Se esegui il comando `kubectl auth can-i --list`, non vedrai alcun permesso di Kubernetes assegnato dalle policy di accesso associate a una voce di accesso per il principale IAM che stai utilizzando quando viene eseguito il comando. Il comando mostra le autorizzazioni di Kubernetes solo se le hai concesse a oggetti `Role` o `ClusterRole` di Kubernetes associati ai nomi dei gruppi o al nome utente specificato per una voce di accesso.
- Se impersoni un utente o un gruppo di Kubernetes durante l'interazione con oggetti di Kubernetes sul cluster, ad esempio utilizzando il comando `kubectl` con `--as` *`username`* o `--as-group` *`group-name`* , stai forzando l'uso dell'autorizzazione RBAC di Kubernetes. Di conseguenza, il principale IAM non dispone di autorizzazioni assegnate da alcuna policy di accesso associata alla voce di accesso. Le uniche autorizzazioni di Kubernetes di cui dispone l'utente o il gruppo che il principale IAM sta impersonando sono quelle di Kubernetes che hai concesso negli oggetti `Role` o `ClusterRole` di Kubernetes associati ai nomi dei gruppi o al nome utente. Per garantire che il

principale IAM abbia le autorizzazioni nelle policy di accesso associate, evitare di impersonare un utente o un gruppo di Kubernetes. Il principale IAM avrà comunque anche tutte le autorizzazioni concesse negli oggetti `Role` o `ClusterRole` di Kubernetes associati ai nomi dei gruppi o al nome utente specificato per la voce di accesso. Per ulteriori informazioni, consulta User impersonation nella documentazione di Kubernetes.

È possibile associare una politica di accesso a una voce di accesso utilizzando Console di gestione AWS o la AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster con una voce di accesso da associare a una policy di accesso.
3. Scegliere la scheda Accesso.
4. Se il tipo di voce di accesso è Standard, puoi associare o dissociare le policy di accesso di Amazon EKS. Se il tipo di voce di accesso è diverso da Standard, questa opzione non è disponibile.
5. Scegliere Associa policy di accesso.
6. Per Nome della policy, seleziona la policy con le autorizzazioni che desideri assegnare al principale IAM. Per visualizzare i permessi inclusi in ciascuna policy, consulta the section called "Rivedere le policy di accesso".
7. Per Ambito di accesso, scegli un ambito di accesso. Se scegli Cluster, le autorizzazioni nella policy di accesso vengono concesse al principale IAM per le risorse in tutti i namespace di Kubernetes. Se scegli Namespace di Kubernetes, puoi scegliere Aggiungi nuovo namespace. Nel campo Namespace che appare, puoi inserire il nome di un namespace di Kubernetes sul cluster. Se desideri che il principale IAM disponga delle autorizzazioni su più namespace, puoi inserirne più di uno.
8. Scegliere Aggiungi policy di accesso.

AWS CLI

1. Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo oppure. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come`yum`,`apt-get`, o Homebrew per macOS sono spesso diverse versioni inferiori all'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e

configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata AWS CloudShell potrebbe inoltre essere inferiore a quella più recente. Per aggiornarla, vedi Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.

2. Visualizza le policy di accesso disponibili.

```
aws eks list-access-policies --output table
```

Di seguito viene riportato un output di esempio.

```
-------------------------------------------------------------------------------------------------
|                                          ListAccessPolicies
                   |
+-----------------------------------------------------------------------------------------------
+
||                                            accessPolicies
                  ||
|+-------------------------------------------------------------------
+------------------------------+|
||                                arn                                |
 name              ||
|+-------------------------------------------------------------------
+------------------------------+|
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSAdminPolicy        |
 AmazonEKSAdminPolicy         ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
 AmazonEKSEditPolicy          ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSViewPolicy         |
 AmazonEKSViewPolicy          ||
|+-------------------------------------------------------------------
+------------------------------+|
```

Per visualizzare i permessi inclusi in ciascuna policy, consulta the section called “Rivedere le policy di accesso”.

3. Visualizza le voci di accesso esistenti. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks list-access-entries --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "accessEntries": [
        "arn:aws: iam::111122223333:role/my-role",
        "arn:aws: iam::111122223333:user/my-user"
    ]
}
```

4. Associa una policy di accesso a una voce di accesso. L'esempio seguente associa la policy di accesso `AmazonEKSViewPolicy` a una voce di accesso. Ogni volta che il ruolo *`my-role`* IAM tenta di accedere agli oggetti Kubernetes sul cluster, Amazon EKS autorizzerà il ruolo a utilizzare le autorizzazioni della policy per accedere agli oggetti Kubernetes solo nei namespace e Kubernetes. *`my-namespace1`* *`my-namespace2`* Sostituiscilo *`my-cluster`* con il nome del tuo cluster, con l'ID del tuo AWS account e *`111122223333`* *`my-role`* con il nome del ruolo IAM per il quale desideri che Amazon EKS autorizzi l'accesso agli oggetti del cluster Kubernetes.

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

Se desideri che il principale IAM disponga delle autorizzazioni a livello di cluster, sostituisci `type=namespace,namespaces=`*`my-namespace1`*`,`*`my-namespace2`* con `type=cluster`. Se desideri associare più policy di accesso alla voce di accesso, esegui il comando più volte, ogni volta con una policy di accesso unica. Ogni policy di accesso associata ha il proprio ambito.

> **Note**
>
> Se in seguito desideri modificare l'ambito di una policy di accesso associata, esegui nuovamente il comando precedente con il nuovo ambito. Ad esempio, se desideri rimuoverli*`my-namespace2`*, esegui nuovamente il comando utilizzando only. `type=namespace,namespaces=`*`my-namespace1`* Se desidera modificare l'ambito da `namespace` a `cluster`, esegui nuovamente il comando utilizzando `type=cluster` e rimuovendo `type=namespace,namespaces=`*`my-namespace1`*`,`*`my-namespace2`* .

5. Determina quali policy di accesso sono associate a una voce di accesso.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-arn
 arn:aws: iam::111122223333:role/my-role
```

Di seguito viene riportato un output di esempio.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws: iam::111122223333:role/my-role",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

Nell'esempio precedente, il principale IAM per questa voce di accesso ha permessi di visualizzazione in tutti i namespace del cluster e permessi di amministratore per due namespace di Kubernetes.

6. Dissociazione di una policy di accesso da una voce di accesso. In questo esempio, la policy `AmazonEKSAdminPolicy` è dissociata da una voce di accesso. Tuttavia, il responsabile IAM

mantiene le autorizzazioni nella politica di `AmazonEKSViewPolicy` accesso per gli oggetti negli *`my-namespace2`* spazi dei nomi *`my-namespace1`* e, poiché tale politica di accesso non è dissociata dalla voce di accesso.

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

Per elencare le policy di accesso disponibili, consultare the section called "Rivedere le policy di accesso".

## Migrazione delle voci **`aws-auth ConfigMap`** esistenti per accedere alle voci

Se hai aggiunto voci a `aws-auth ConfigMap` sul tuo cluster, ti consigliamo di creare voci di accesso per quelle esistenti in `aws-auth ConfigMap`. Dopo aver creato le voci di accesso, puoi rimuovere le voci da `ConfigMap`. Non puoi associare le policy di accesso alle voci presenti in `aws-auth ConfigMap`. Se desideri associare le politiche di accesso ai tuoi principali IAM, crea delle voci di accesso.

> **Important**
>
> - Quando un cluster è in modalità di autenticazione `API_AND_CONFIGMAP` ed è presente una mappatura per lo stesso ruolo IAM sia in `aws-auth ConfigMap` che nelle voci di accesso, il ruolo utilizzerà la mappatura della voce di accesso per l'autenticazione. Le voci di accesso hanno la precedenza sulle voci `ConfigMap` dello stesso principale IAM.
> - Prima di rimuovere le voci `aws-auth ConfigMap` esistenti create da Amazon EKS per un gruppo di nodi gestiti o un profilo Fargate nel tuo cluster, ricontrolla se nel tuo cluster Amazon EKS esistono le voci di accesso corrette per quelle risorse specifiche. Se rimuovi le voci che Amazon EKS ha creato in `ConfigMap` senza avere le voci di accesso equivalenti, il cluster non funzionerà correttamente.

### Prerequisiti

- Familiarità con le voci di accesso e le policy di accesso. Per ulteriori informazioni, consulta the section called "Voci di accesso" e the section called "Associare le policy di accesso".

- Un cluster esistente con una versione della piattaforma corrispondente o successiva alle versioni elencate nei Prerequisiti dell'argomento the section called "Voci di accesso".
- La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
- Le autorizzazioni di Kubernetes per modificare `aws-auth ConfigMap` nel namespace `kube-system`.
- Un ruolo o utente di gestione delle AWS identità e degli accessi con le seguenti autorizzazioni: `CreateAccessEntry` e. `ListAccessEntries` Per ulteriori informazioni, consulta Actions defined by Amazon Elastic Kubernetes Service nella Documentazione di riferimento per l'autorizzazione ai servizi.

### `eksctl`

1. Verifica le voci esistenti in `aws-auth ConfigMap`. Sostituisci *`my-cluster`* con il nome del cluster.

```
eksctl get iamidentitymapping --cluster my-cluster
```

Di seguito viene riportato un output di esempio.

```
ARN
            USERNAME                                      GROUPS
                        ACCOUNT
arn:aws: iam::111122223333:role/EKS-my-cluster-Admins
            Admins                                        system:masters
arn:aws: iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
            my-namespace-Viewers                          Viewers
arn:aws: iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
arn:aws: iam::111122223333:user/my-user
            my-user
arn:aws: iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws: iam::111122223333:role/EKS-my-cluster-managed-ng
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
```

2. the section called "Creare voci di accesso" per tutte le voci `ConfigMap` che hai creato e che sono state restituite nell'output precedente. Quando crei le voci di accesso, assicurati di specificare gli stessi valori per `ARN`, `USERNAME`, `GROUPS` e `ACCOUNT` restituiti nell'output. Nell'output di esempio, dovresti creare voci di accesso per tutte le voci tranne le ultime due, poiché tali voci sono state create da Amazon EKS per un profilo Fargate e un gruppo di nodi gestito.
3. Elimina le voci da `ConfigMap` per tutte le voci di accesso che hai creato. Se non elimini la voce da `ConfigMap`, le impostazioni per la voce di accesso per l'ARN principale IAM hanno la precedenza sulla voce `ConfigMap`. Sostituiscilo *`111122223333`* con l'ID del tuo AWS account e *`EKS-my-cluster-my-namespace-Viewers`* con il nome del ruolo nella voce del tuo`ConfigMap`. Se la voce che stai rimuovendo è per un utente IAM, anziché per un ruolo IAM, `role` sostituiscila con `user` e *`EKS-my-cluster-my-namespace-Viewers`* con il nome utente.

```
eksctl delete iamidentitymapping --arn arn:aws: iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

## Rivedere le autorizzazioni della policy di accesso

Le policy di accesso includono `rules` che contengono `verbs` di Kubernetes (autorizzazioni) e `resources`. Le policy di accesso non includono le autorizzazioni o le risorse IAM. Analogamente agli oggetti `Role` e `ClusterRole` di Kubernetes, le policy di accesso includono solo `allow` `rules`. Non puoi modificare il contenuto di una policy di accesso. Non puoi creare policy di accesso personalizzate. Se le autorizzazioni nelle policy di accesso non soddisfano le tue esigenze, crea oggetti RBAC di Kubernetes e specificare nomi dei gruppi per le voci di accesso. Per ulteriori informazioni, consulta the section called "Creare voci di accesso". Le autorizzazioni contenute nelle policy di accesso sono simili a quelle nei ruoli del cluster rivolti agli utenti di Kubernetes. Per ulteriori informazioni, consulta User-facing i ruoli nella documentazione di Kubernetes.

Elencare tutte le policy

Utilizza una delle politiche di accesso elencate in questa pagina o recupera un elenco di tutte le politiche di accesso disponibili utilizzando la CLI: AWS

```
aws eks list-access-policies
```

L'output previsto dovrebbe essere simile al seguente (abbreviato per brevità):

```
{
    "accessPolicies": [
```

```
        {
            "name": "AmazonAIOpsAssistantPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonAIOpsAssistantPolicy"
        },
        {
            "name": "AmazonARCRegionSwitchScalingPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/
AmazonARCRegionSwitchScalingPolicy"
        },
        {
            "name": "AmazonEKSAdminPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy"
        },
        {
            "name": "AmazonEKSAdminViewPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy"
        },
        {
            "name": "AmazonEKSAutoNodePolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy"
        }
        // Additional policies omitted
    ]
}
```

AmazonEKSAdminPolicy

Questa policy di accesso include autorizzazioni che concedono a un principale IAM la maggior parte delle autorizzazioni per le risorse. Quando è associato a una voce di accesso, il relativo ambito di accesso è in genere uno o più namespace di Kubernetes. Se invece desideri che un principale IAM disponga dell'accesso di amministratore a tutte le risorse del cluster, allora associa la policy di accesso the section called "AmazonEKSClusterAdminPolicy" alla voce di accesso.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| `apps` | `daemonsets` , `deployments` , `deployments/rollback` , `deployments/scale` , `replicase` | `create`, `delete`, `deletecollection` , `patch`, `update` |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| | ts , replicasets/ scale, statefulsets , statefulsets/scale | |
| apps | controllerrevisions , daemonsets , daemonset s/status , deploymen ts , deployments/scale , deployments/status , replicasets , replicase ts/scale , replicase ts/status , statefuls ets , statefulsets/ scale , statefulsets/ status | get, list, watch |
| authorization.k8s.io | localsubjectaccess reviews | create |
| autoscaling | horizontalpodautos calers | create, delete, deletecol lection , patch, update |
| autoscaling | horizontalpodautos calers , horizonta lpodautoscalers/st atus | get, list, watch |
| batch | cronjobs, jobs | create, delete, deletecol lection , patch, update |
| batch | cronjobs, cronjobs/ status , jobs, jobs/stat us | get, list, watch |
| discovery.k8s.io | endpointslices | get, list, watch |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| extensions | daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale | create, delete, deletecollection , patch, update |
| extensions | daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale | get, list, watch |
| networking.k8s.io | ingresses , ingresses/status , networkpolicies | get, list, watch |
| networking.k8s.io | ingresses , networkpolicies | create, delete, deletecollection , patch, update |
| policy | poddisruptionbudgets | create, delete, deletecollection , patch, update |
| policy | poddisruptionbudgets , poddisruptionbudgets/status | get, list, watch |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| rbac.authorization .k8s.io | rolebindings , roles | create, delete, deletecol lection , get, list, patch, update, watch |
| | configmaps , endpoints , persistentvolumecl aims , persisten tvolumeclaims/stat us , pods, replicati oncontrollers , replicationcontrol lers/scale , serviceac counts , services, services/status | get, list, watch |
| | pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy | get, list, watch |
| | configmaps , events, persistentvolumecl aims , replicati oncontrollers , replicationcontrol lers/scale , secrets, serviceaccounts , services, services/ proxy | create, delete, deletecol lection , patch, update |
| | pods, pods/attach , pods/exec , pods/port forward , pods/proxy | create, delete, deletecol lection , patch, update |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| | `serviceaccounts` | `impersonate` |
| | `bindings`, `events`, `limitranges` , `namespaces/status` , `pods/log`, `pods/status` , `replicationcontrollers/status` , `resourcequotas` , `resourcequotas/status` | `get`, `list`, `watch` |
| | `namespaces` | `get`, `list`, `watch` |

AmazonEKSClusterAdminPolicy

Questa policy di accesso include le autorizzazioni che concedono a un principale IAM l'accesso di amministratore per un cluster. Quando è associato a una voce di accesso, il relativo ambito di accesso è in genere il cluster, anziché un namespace di Kubernetes. Se desideri che un principale IAM abbia un ambito amministrativo più limitato, valuta invece la possibilità di associare la policy di accesso the section called "AmazonEKSAdminPolicy" alla voce di accesso.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy`

| Gruppi di API Kubernetes | nonResourceURLs di Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|---|
| * | | * | * |
| | * | | * |

AmazonEKSAdminViewPolicy

Questa policy di accesso include autorizzazioni che concedono a un IAM l'accesso principale a list/view tutte le risorse in un cluster. Questo include Kubernetes Secrets.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy`

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| * | * | get, list, watch |

AmazonEKSEditPolicy

Questa policy di accesso include autorizzazioni che consentono a un principale IAM di modificare la maggior parte delle risorse di Kubernetes.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSEditPolicy`

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| apps | daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale | create, delete, deletecollection , patch, update |
| apps | controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status | get, list, watch |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| autoscaling | horizontalpodautoscalers , horizontalpodautoscalers/status | get, list, watch |
| autoscaling | horizontalpodautoscalers | create, delete, deletecollection , patch, update |
| batch | cronjobs, jobs | create, delete, deletecollection , patch, update |
| batch | cronjobs, cronjobs/status , jobs, jobs/status | get, list, watch |
| discovery.k8s.io | endpointslices | get, list, watch |
| extensions | daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale | create, delete, deletecollection , patch, update |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| extensions | daemonsets , daemonset s/status , deploymen ts , deployments/scale , deployments/status , ingresses , ingresses /status , networkpo licies , replicasets , replicasets/scale , replicasets/status , replicationcontrol lers/scale | get, list, watch |
| networking.k8s.io | ingresses , networkpo licies | create, delete, deletecol lection , patch, update |
| networking.k8s.io | ingresses , ingresses /status , networkpo licies | get, list, watch |
| policy | poddisruptionbudgets | create, delete, deletecol lection , patch, update |
| policy | poddisruptionbudgets , poddisruptionbudge ts/status | get, list, watch |
| | namespaces | get, list, watch |
| | pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy | get, list, watch |
| | serviceaccounts | impersonate |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| | pods, pods/attach , pods/exec , pods/port forward , pods/proxy | create, delete, deletecol lection , patch, update |
| | configmaps , events, persistentvolumecl aims , replicati oncontrollers , replicationcontrol lers/scale , secrets, serviceaccounts , services, services/ proxy | create, delete, deletecol lection , patch, update |
| | configmaps , endpoints , persistentvolumecl aims , persisten tvolumeclaims/stat us , pods, replicati oncontrollers , replicationcontrol lers/scale , serviceac counts , services, services/status | get, list, watch |
| | bindings, events, limitranges , namespace s/status , pods/log, pods/status , replicati oncontrollers/stat us , resourcequotas , resourcequotas/sta tus | get, list, watch |

AmazonEKSViewPolicy

Questa policy di accesso include autorizzazioni che consentono a un principale IAM di visualizzare la maggior parte delle risorse di Kubernetes.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy`

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| apps | controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status | get, list, watch |
| autoscaling | horizontalpodautoscalers , horizontalpodautoscalers/status | get, list, watch |
| batch | cronjobs, cronjobs/status , jobs, jobs/status | get, list, watch |
| discovery.k8s.io | endpointslices | get, list, watch |
| extensions | daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpo | get, list, watch |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| | licies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale | |
| networking.k8s.io | ingresses , ingresses/status , networkpolicies | get, list, watch |
| policy | poddisruptionbudgets , poddisruptionbudgets/status | get, list, watch |
| | configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status | get, list, watch |
| | bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status | get, list, watch |
| | namespaces | get, list, watch |

AmazonEKSPodIdentityPolicy

Questa politica di accesso include autorizzazioni che consentono a un responsabile IAM di elencare i pod Kubernetes. Questa policy viene utilizzata internamente dal ruolo collegato al servizio Amazon EKS.

> **Note**
>
> Questa policy è progettata solo per i ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSPodIdentityPolicy`

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| | `pods` | `list` |

AmazonEKSSecretAdminPolicy

Questa policy di accesso include autorizzazioni che consentono a un responsabile IAM l'accesso completo a Kubernetes Secrets. https://kubernetes.io/docs/concepts/configuration/secret/

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSSecretAdminPolicy`

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| | `secrets` | `*` |

AmazonEKSSecretReaderPolicy

Questa policy di accesso include autorizzazioni che consentono a un responsabile IAM di leggere Kubernetes Secrets. https://kubernetes.io/docs/concepts/configuration/secret/

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy`

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| | `secrets` | `get`, `list`, `watch` |

AmazonEKSAutoNodePolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy`

Questa policy include le seguenti autorizzazioni che consentono ai componenti di Amazon EKS di completare le attività seguenti:

- `kube-proxy`: monitorare gli endpoint e i servizi di rete e gestire gli eventi correlati. Ciò consente la funzionalità proxy di rete a livello di cluster.
- `ipamd`— Gestisci le risorse di rete AWS VPC e le interfacce di rete per container (CNI). Ciò consente al daemon di gestione degli indirizzi IP di gestire il networking dei pod.
- `coredns`: accedere a risorse di rilevamento servizi come endpoint e servizi. Ciò consente la risoluzione DNS all'interno del cluster.
- `ebs-csi-driver`: lavorare con risorse relative all'archiviazione per i volumi Amazon EBS. Ciò consente il provisioning e il collegamento dinamico di volumi persistenti.
- `neuron`— Monitora nodi e pod per i dispositivi Neuron. AWS Ciò consente la gestione degli acceleratori AWS Inferentia e Trainium.
- `node-monitoring-agent`: accedere alla diagnostica dei nodi e agli eventi. Ciò consente il monitoraggio dell'integrità del cluster e la raccolta della diagnostica.

Ogni componente utilizza un account di servizio dedicato ed è limitato alle sole autorizzazioni necessarie per la sua funzione specifica.

Se si specifica manualmente un ruolo IAM del nodo in a NodeClass, è necessario creare una voce di accesso che associ il nuovo ruolo IAM del nodo a questa policy di accesso.

AmazonEKSBlockStoragePolicy

> **Note**
>
> Questa policy è designata solo per i ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStoragePolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per l'elezione e il coordinamento dei leader per le operazioni di archiviazione:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di archiviazione EKS di coordinare le proprie attività all'interno del cluster attraverso un meccanismo di elezione dei leader.

La policy si applica a specifiche risorse di leasing utilizzate dai componenti di archiviazione EKS per prevenire conflitti di accesso ad altre risorse di coordinamento del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di storage a blocchi.

AmazonEKSLoadBalancingPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingPolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per le elezioni per i leader per il bilanciamento del carico:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di bilanciamento del carico EKS di coordinare le attività su più repliche eleggendo un leader.

La policy riguarda specificamente il bilanciamento del carico delle risorse di leasing per garantire un coordinamento adeguato impedendo al contempo l'accesso ad altre risorse di leasing del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di rete.

AmazonEKSNetworkingPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingPolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per le elezioni dei leader per le reti:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di rete EKS di coordinare le attività di allocazione degli indirizzi IP eleggendo un leader.

La policy riguarda specificamente le reti delle risorse di leasing per garantire un coordinamento adeguato impedendo al contempo l'accesso ad altre risorse di leasing del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di rete.

AmazonEKSComputePolicy

> **Note**
>
> Questa policy è progettata solo per i ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputePolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per l'elezione dei leader per le operazioni di elaborazione:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di elaborazione EKS di coordinare le attività di scalabilità dei nodi eleggendo un leader.

La policy riguarda specificamente la gestione delle risorse di elaborazione in leasing e consente al contempo l'accesso di base in lettura (`get`, `watch`) a tutte le risorse di leasing del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di gestione di elaborazione.

AmazonEKSBlockStorageClusterPolicy

> **Note**
>
> Questa policy è progettata solo per i ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStorageClusterPolicy`

Questa policy concede le autorizzazioni necessarie per la funzionalità di storage a blocchi della modalità automatica di Amazon EKS. Consente una gestione efficiente delle risorse di archiviazione a blocchi all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione del driver CSI:

- Creare, leggere, aggiornare ed eliminare i driver CSI, in particolare per l'archiviazione a blocchi.

Gestione dei volumi:

- Elencare, guardare, creare, aggiornare, applicare patch ed eliminare i volumi persistenti.
- Elencare, guardare e aggiornare le dichiarazioni di volume persistenti.
- Applicare patch agli stati di dichiarazione di volume persistente.

Interazione tra nodi e pod:

- Leggere le informazioni su nodi e pod.
- Gestire gli eventi relativi alle operazioni di archiviazione.

Classi e attributi di archiviazione:

- Leggere le classi di archiviazione e i nodi CSI.
- Leggere le classi di attributi del volume.

Allegati dei volumi:

- Elencare, guardare e modificare gli allegati del volume e il relativo stato.

Operazioni di snapshot:

- Gestire snapshot di volume, contenuti di snapshot e classi di snapshot.
- Gestire le operazioni per gli snapshot dei gruppi di volumi e le risorse correlate.

Questa policy è progettata per supportare la gestione completa dell'archiviazione a blocchi all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per varie operazioni, tra cui il provisioning, il collegamento, il ridimensionamento e la creazione di snapshot di volumi di archiviazione a blocchi.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di storage a blocchi.

AmazonEKSComputeClusterPolicy

> **Note**
>
> Questa policy è progettata solo per i ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputeClusterPolicy`

Questa policy concede le autorizzazioni necessarie per la funzionalità di gestione dell'elaborazione della modalità automatica di Amazon EKS. Consente l'orchestrazione e la scalabilità efficienti delle risorse di elaborazione all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione dei nodi:

- Crea, leggi, aggiorna, elimina e gestisci lo stato di e. NodePools NodeClaims

- Gestione NodeClasses, inclusa la creazione, la modifica e l'eliminazione.

Pianificazione e gestione delle risorse:

- Accesso in lettura a pod, nodi, volumi persistenti, attestazioni di volume persistenti, controller di replica e namespace.
- Accesso in lettura alle classi di archiviazione, ai nodi CSI e agli allegati dei volumi.
- Elencare e controllare implementazioni, set di daemon, set di repliche e set stateful.
- Leggere i budget di interruzione dei pod.

Gestione degli eventi:

- Creare, leggere e gestire gli eventi del cluster.

Revoca del provisioning dei nodi ed espulsione dei pod:

- Aggiornare, applicare patch ed eliminare i nodi.
- Creare espulsioni dei pod ed eliminarli quando necessario.

Gestione di definizione di risorse personalizzate (CRD):

- Creare nuove CRD.
- Gestisci CRD specifici relativi alla gestione dei nodi (NodeClasses, NodePools NodeClaims, e NodeDiagnostics).

Questa policy è progettata per supportare la gestione completa dell'elaborazione all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per varie operazioni, tra cui il provisioning dei nodi, la pianificazione, la scalabilità e l'ottimizzazione delle risorse.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di gestione di elaborazione.

AmazonEKSLoadBalancingClusterPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingClusterPolicy`

Questa policy concede le autorizzazioni necessarie per la funzionalità di bilanciamento del carico della modalità automatica di Amazon EKS. Consente una configurazione e una gestione efficiente delle risorse di bilanciamento del carico all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione di eventi e risorse:

- Creare e applicare patch agli eventi.
- Accesso in lettura a pod, nodi, endpoint e namespace.
- Aggiornare gli stati dei pod.

Gestione dei servizi e degli ingressi:

- Gestione completa dei servizi e dei relativi stati.
- Controllo completo sugli ingressi e sui relativi stati.
- Accesso in lettura alle sezioni degli endpoint e alle classi di ingresso.

Associazioni per gruppi di destinazione:

- Creare e modificare le associazioni per gruppi di destinazione e i relativi stati.
- Accesso in lettura ai parametri della classe di ingresso.

Gestione di definizione di risorse personalizzate (CRD):

- Creare e leggere tutte le CRD.
- Gestione specifica delle CRD targetgroupbindings.eks.amazonaws.com e ingressclassparams.eks.amazonaws.com.

Configurazione Webhook:

- Creare e leggere configurazioni webhook mutanti e convalidanti.
- Gestire la configurazione eks-load-balancing-webhook.

Questa policy è progettata per supportare la gestione completa del bilanciamento del carico all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per varie

operazioni, tra cui l'esposizione dei servizi, il routing in ingresso e l'integrazione con AWS i servizi di bilanciamento del carico.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di bilanciamento del carico.

AmazonEKSNetworkingClusterPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingClusterPolicy`

AmazonEKSNetworkingClusterPolicy

Questa policy concede le autorizzazioni necessarie per la funzionalità di rete della modalità automatica di Amazon EKS. Consente una configurazione e una gestione efficiente delle risorse di rete all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione di nodi e pod:

- Accesso in lettura e relativo stato. NodeClasses
- Accesso in lettura NodeClaims e relativo stato.
- Accesso in lettura ai pod.

Gestione dei nodi CNI:

- Autorizzazioni per CNINodes e relativi stati, tra cui creazione, lettura, aggiornamento, eliminazione e applicazione di patch.

Gestione di definizione di risorse personalizzate (CRD):

- Creare e leggere tutte le CRD.
- Gestione specifica (aggiornamento, applicazione di patch, eliminazione) della CRD cninodes.eks.amazonaws.com.

Gestione degli eventi:

- Creare e applicare patch agli eventi.

Questa policy è progettata per supportare la gestione completa della rete all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per varie operazioni, tra cui la configurazione di rete dei nodi, la gestione della CNI (Container Network Interface) e la relativa gestione personalizzata delle risorse.

La policy consente ai componenti di rete di interagire con le risorse relative ai nodi, gestire le configurazioni dei CNI-specific nodi e gestire risorse personalizzate fondamentali per le operazioni di rete nel cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di rete.

AmazonEKSHybridPolicy

> **Note**
>
> Questa policy è designata solo per i ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

Questa policy di accesso include le autorizzazioni che concedono a EKS l'accesso di EKS ai nodi di un cluster. Quando è associato a una voce di accesso, il relativo ambito di accesso è in genere il cluster, anziché un namespace di Kubernetes. Questa policy viene utilizzata da Amazon EKS Hybrid Nodes.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSHybridPolicy`

| Gruppi di API Kubernetes | nonResourceURLs di Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|---|
| * | | nodes | list |

AmazonEKSClusterInsightsPolicy

> **Note**
>
> Questa policy è progettata solo per i ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterInsightsPolicy`

Questa policy concede le autorizzazioni di sola lettura alla funzionalità di approfondimenti del cluster di Amazon EKS. La policy include le seguenti autorizzazioni:

Accesso ai nodi: - Elencare e visualizzare i nodi del cluster - Leggere le informazioni sullo stato dei nodi

DaemonSet Accesso: - Accesso in lettura alla configurazione di kube-proxy

Questa policy viene gestita automaticamente dal servizio EKS per gli approfondimenti del cluster. Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

AWSBackupFullAccessPolicyForBackup

ARN: `arn:aws: eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForBackup`

AWSBackupFullAccessPolicyForBackup

Questa policy concede le autorizzazioni necessarie a AWS Backup per gestire e creare backup del cluster EKS. Questa policy include le seguenti autorizzazioni:

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| * | * | `list`, `get` |

AWSBackupFullAccessPolicyForRestore

ARN: `arn:aws: eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForRestore`

AWSBackupFullAccessPolicyForRestore

Questa politica concede le autorizzazioni necessarie a AWS Backup per gestire e ripristinare i backup del cluster EKS. Questa policy include le seguenti autorizzazioni:

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| `*` | `*` | `list`, `get`, `create` |
| `rbac.authorization.k8s.io` | `roles`, `clusterroles` | `create`, `update`, `escalate`, `bind` |

Politica Amazon EKSack

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSACKPolicy`

Questa policy concede le autorizzazioni necessarie affinché la funzionalità AWS Controllers for Kubernetes (ACK) gestisca le risorse di Kubernetes. AWS La policy include le seguenti autorizzazioni:

Gestione personalizzata delle risorse ACK:

- Accesso completo a tutte le risorse personalizzate del servizio ACK in oltre 50 AWS servizi tra cui S3, RDS, DynamoDB, Lambda, EC2 e altri.
- Crea, leggi, aggiorna ed elimina le definizioni di risorse personalizzate ACK.

Accesso al namespace:

- Accesso in lettura ai namespace per l'organizzazione delle risorse.

Elezione del leader:

- Crea e leggi i contratti di coordinamento per l'elezione dei leader.
- Aggiorna ed elimina i leasing specifici dei controller di servizio ACK.

Gestione degli eventi:

- Crea e applica patch agli eventi per le operazioni ACK.

Questa policy è progettata per supportare una gestione completa AWS delle risorse tramite le API Kubernetes. Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM della funzionalità fornito al momento della creazione della funzionalità ACK.

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| | `namespaces` | `get`, `watch`, `list` |
| `services.k8s.aws` , `acm.services.k8s.aws` , `acmpca.services.k8s.aws` , `apigateway.services.k8s.aws` , `apigatewayv2.services.k8s.aws` , `applicationautoscaling.services.k8s.aws` , `athena.services.k8s.aws` , `bedrock.services.k8s.aws` , `bedrockagent.services.k8s.aws` , `bedrockagentcorecontrol.services.k8s.aws` , `cloudfront.services.k8s.aws` , `cloudtrail.services.k8s.aws` , `cloudwatch.services.k8s.aws` , `cloudwatchlogs.services.k8s.aws` , `codeartifact.services.k8s.a` | * | * |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
| --- | --- | --- |
| ws , cognitoidentityprovider.services.k8s.aws , documentdb.services.k8s.aws , dynamodb.services.k8s.aws , ec2.services.k8s.aws , ecr.services.k8s.aws , ecrpublic.services.k8s.aws , ecs.services.k8s.aws , efs.services.k8s.aws , eks.services.k8s.aws , elasticache.services.k8s.aws , elbv2.services.k8s.aws , emrcontainers.services.k8s.aws , eventbridge.services.k8s.aws , iam.services.k8s.aws , kafka.services.k8s.aws , keyspaces.services.k8s.aws , kinesis.services.k8s.aws , kms.services.k8s.aws , lambda.services.k8s.aws , memorydb.services.k8s.aws , mq.services.k8s.aws , networkfirewall.services.k8s.aws , | | |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| opensearchservice.services.k8s.aws , organizations.services.k8s.aws , pipes.services.k8s.aws , prometheusservice.services.k8s.aws , ram.services.k8s.aws , rds.services.k8s.aws , recyclebin.services.k8s.aws , route53.services.k8s.aws , route53resolver.services.k8s.aws , s3.services.k8s.aws , s3control.services.k8s.aws , sagemaker.services.k8s.aws , secretsmanager.services.k8s.aws , ses.services.k8s.aws , sfn.services.k8s.aws , sns.services.k8s.aws , sqs.services.k8s.aws , ssm.services.k8s.aws , wafv2.services.k8s.aws | | |
| coordination.k8s.io | leases | create, get, list, watch |
| coordination.k8s.io | leases(solo contratti di locazione di controller di servizio ACK specifici) | delete, update, patch |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| | `events` | `create`, `patch` |
| `apiextensions.k8s.io` | `customresourcedefinitions` | `*` |

AmazonEKSArgoCDClusterPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy`

Questa policy concede le autorizzazioni a livello di cluster necessarie per consentire alla funzionalità Argo CD di rilevare risorse e gestire oggetti con ambito di cluster. La policy include le seguenti autorizzazioni:

Gestione dello spazio dei nomi:

- Crea, leggi, aggiorna ed elimina namespace per la gestione degli spazi dei nomi delle applicazioni.

Gestione personalizzata delle definizioni delle risorse:

- Gestisci i CD-specific CRD Argo (Applicazioni, AppProjects, ApplicationSets).

Scoperta delle API:

- Accesso in lettura agli endpoint dell'API Kubernetes per l'individuazione delle risorse.

Questa policy è progettata per supportare le operazioni Argo CD a livello di cluster, inclusa la gestione dei namespace e l'installazione di CRD. Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM della funzionalità fornito al momento della creazione della funzionalità Argo CD.

| Gruppi di API Kubernetes | nonResourceURLs di Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|---|
| | | namespaces | create, get, update, patch, delete |
| apiextensions.k8s.io | | customresourcedefinitions | create |
| apiextensions.k8s.io | | customresourcedefinitions (solo CD CRD Argo) | get, update, patch, delete |
| | /api, /api/*, /apis, /apis/* | | get |

AmazonEKSArgoCDPolicy

ARN: arn:aws: eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy

Questa policy concede le autorizzazioni a livello di namespace necessarie per consentire alla funzionalità Argo CD di distribuire e gestire le applicazioni. La policy include le seguenti autorizzazioni:

Gestione segreta:

- Accesso completo ai segreti per le credenziali Git e i segreti del cluster.

ConfigMap Accesso:

- Accesso in lettura per ConfigMaps inviare avvisi se i clienti tentano di utilizzare un CD Argo non supportato. ConfigMaps

Gestione degli eventi:

- Leggi e crea eventi per il monitoraggio del ciclo di vita delle applicazioni.

Gestione delle risorse Argo CD:

- Accesso completo alle applicazioni, ApplicationSets, e AppProjects.
- Gestisci i finalizzatori e lo stato delle risorse Argo CD.

Questa policy è progettata per supportare le operazioni sui CD Argo a livello di namespace, inclusa la distribuzione e la gestione delle applicazioni. Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM della funzionalità fornito al momento della creazione della funzionalità Argo CD, con ambito nello spazio dei nomi Argo CD.

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| | `secrets` | `*` |
| | `configmaps` | `get`, `list`, `watch` |
| | `events` | `get`, `list`, `watch`, `patch`, `create` |
| `argoproj.io` | `applications` , `applications/finalizers` , `applications/status` , `applicationsets` , `applicationsets/finalizers` , `applicationsets/status` , `appprojects` , `appprojects/finalizers` , `appprojects/status` | `*` |

Politica Amazon EKSKRO

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSKROPolicy`

Questa policy concede le autorizzazioni necessarie alla funzionalità kro (Kube Resource Orchestrator) di creare e gestire API Kubernetes personalizzate. La policy include le seguenti autorizzazioni:

Gestione delle risorse kro:

- Accesso completo a tutte le risorse kro, incluse le istanze ResourceGraphDefinitions di risorse personalizzate.

Gestione personalizzata delle definizioni delle risorse:

- Crea, leggi, aggiorna ed elimina CRD per le API personalizzate definite da. ResourceGraphDefinitions

Elezione del leader:

- Crea e leggi i contratti di coordinamento per l'elezione dei leader.
- Aggiorna ed elimina il contratto di locazione del controller kro.

Gestione degli eventi:

- Crea e correggi gli eventi per le operazioni kro.

Questa policy è progettata per supportare la composizione completa delle risorse e la gestione personalizzata delle API tramite kro. Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM della funzionalità fornito al momento della creazione della funzionalità kro.

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizzazioni) |
|---|---|---|
| `kro.run` | `*` | `*` |
| `apiextensions.k8s.io` | `customresourcedefinitions` | `*` |
| `coordination.k8s.io` | `leases` | `create`, `get`, `list`, `watch` |

| Gruppi di API Kubernetes | Risorse di Kubernetes | Verbi di Kubernetes (autorizz azioni) |
|---|---|---|
| `coordination.k8s.io` | `leases`(solo kro controller in lease) | `delete`, `update`, `patch` |
| | `events` | `create`, `patch` |

Aggiornamenti della policy di accesso

Visualizza i dettagli sugli aggiornamenti alle policy di accesso, da quando sono stati introdotti. Per gli avvisi automatici sulle modifiche apportate a questa pagina, iscriviti al feed RSS in Cronologia dei documenti.

| Modifica | Descrizione | Data |
|---|---|---|
| Aggiungi politiche per EKS Capabilities | Pubblica`AmazonEKS ACKPolicy` , `AmazonEKS ArgoCDClusterPolic y` `AmazonEKSArgoCDPol icy` , e `AmazonEKS KROPolicy` per gestire EKS Capabilities | 22 novembre 2025 |
| Add `AmazonEKSSecretRea derPolicy` | Aggiungi una nuova politica per l'accesso in sola lettura ai segreti | 6 novembre 2025 |
| Aggiungi una policy per gli approfondimenti sui cluster di EKS | Pubblica `AmazonEKS ClusterInsightsPol icy` | 2 dicembre 2024 |
| Aggiungi policy per Amazon EKS Hybrid | Pubblica `AmazonEKS HybridPolicy` | 2 dicembre 2024 |
| Aggiungi policy per la modalità automatica di Amazon EKS | Queste policy di accesso autorizzano il Cluster IAM Role e il Node IAM Role a | 2 dicembre 2024 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| | chiamare le API Kubernetes. AWS le utilizza per automatizzare le attività di routine per le risorse di archiviazione, elaborazione e rete. | |
| Add `AmazonEKSAdminViewPolicy` | Aggiungi una nuova policy per un accesso esteso alle visualizzazioni, incluse risorse come Secrets. | 23 aprile 2024 |
| Policy di accesso introdotte. | Amazon EKS ha introdotto policy di accesso. | 29 maggio 2023 |

## Cambia la modalità di autenticazione per utilizzare le voci di accesso

Per iniziare a utilizzare le voci di accesso, è necessario modificare la modalità di autenticazione del cluster nelle modalità `API_AND_CONFIG_MAP` o `API`. Questa operazione aggiunge l'API per le voci di accesso.

Console AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster in cui desideri creare una voce di accesso.
3. Scegli la scheda Accesso.
4. Modalità di autenticazione mostra la modalità corrente di autenticazione del cluster. Se la modalità mostra API EKS, puoi già aggiungere voci di accesso e puoi saltare i passaggi rimanenti.
5. Scegli Gestisci accesso.
6. Per Modalità di autenticazione del cluster, seleziona una modalità con l'API EKS. Nota che non è possibile ripristinare la modalità di autenticazione in una modalità che rimuove l'API EKS e le voci di accesso.
7. Scegli Save changes (Salva modifiche). Amazon EKS inizia ad aggiornare il cluster, lo stato del cluster cambia in In aggiornamento e la modifica è registrata nella scheda Cronologia degli aggiornamenti.

8. Attendi che lo stato del cluster ritorni su Attivo. Quando il cluster è Attivo, è possibile seguire i passaggi indicati in the section called "Creare voci di accesso" per aggiungere l'accesso al cluster per i principali IAM.

CLI AWS

1. Installa AWS CLI come descritto in Installazione nella Guida per l'utente dell'interfaccia a riga di comando AWS.
2. Esegui il comando seguente. Sostituisci *my-cluster* con il nome del tuo cluster. Se desideri disabilitare il metodo `ConfigMap` in modo permanente, sostituisci `API_AND_CONFIG_MAP` con `API`.

   Amazon EKS inizia ad aggiornare il cluster, lo stato del cluster cambia in IN AGGIORNAMENTO e la modifica è registrata negli  aggiornamenti dell'elenco eks aws .

   ```
   aws eks update-cluster-config --name my-cluster --access-config
    authenticationMode=API_AND_CONFIG_MAP
   ```
3. Attendi che lo stato del cluster ritorni su Attivo. Quando il cluster è Attivo, è possibile seguire i passaggi indicati in the section called "Creare voci di accesso" per aggiungere l'accesso al cluster per i principali IAM.

Versione della piattaforma richiesta

Per utilizzare le voci di accesso, il cluster deve avere una versione della piattaforma uguale o successiva a quella elencata nella tabella seguente o una versione Kubernetes successiva a quelle elencate nella tabella. Se la tua versione di Kubernetes non è elencata, tutte le versioni della piattaforma supportano le voci di accesso.

| Versione di Kubernetes 1.15 | Versione della piattaforma |
|---|---|
| Non elencata | Tutte supportate |
| `1.30` | `eks.2` |
| `1.29` | `eks.1` |
| `1.28` | `eks.6` |

Per ulteriori informazioni, consulta Versioni della piattaforma .

## Creazione di voci di accesso

Prima di creare voci di accesso, considera quanto segue:

- Una modalità di autenticazione impostata in modo corretto. Consulta the section called "Modalità di autenticazione".
- Una voce di accesso include il nome della risorsa Amazon (ARN) di un principale (e solo di uno) IAM esistente. Un principale IAM non può essere incluso in più di una voce di accesso. Considerazioni aggiuntive per l'ARN specificato:
  - Le best practice IAM consigliano di accedere al cluster utilizzando ruoli IAM con credenziali a breve termine, anziché utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella IAM User Guide.
  - Se l'ARN è per un ruolo IAM, può includere un percorso. Gli ARN nelle voci `aws-auth ConfigMap`, non possono includere un percorso. Ad esempio, il tuo ARN può essere `arn:aws: iam::<111122223333>:role/<development/apps/my-role>` o `arn:aws: iam::<111122223333>:role/<my-role>`.
  - Se il tipo di voce di accesso è diverso da `STANDARD` (vedi la successiva considerazione sui tipi), l'ARN deve trovarsi nello stesso AWS account in cui si trova il cluster. Se il tipo è`STANDARD`, l'ARN può essere nello stesso account o in un altro AWS account rispetto all'account in cui si trova il cluster.
  - Non è possibile modificare il principale IAM dopo aver creato la voce di accesso.
  - Se elimini il principale IAM con questo ARN, la voce di accesso non viene eliminata automaticamente. Ti consigliamo di eliminare la voce di accesso con un ARN per un principale IAM che elimini. Se non cancelli la voce di accesso e in futuro ricrei il principale IAM, anche se ha lo stesso ARN, la voce di accesso non funzionerà. Questo perché, anche se l'ARN è lo stesso per l'entità IAM ricreata, l'`roleID`or `userID` (si può vedere con il comando `aws sts get-caller-identity` AWS CLI) è diverso per l'entità IAM ricreata rispetto all'entità IAM originale. Anche se non visualizzi il `roleID` o il `userID` del principale IAM per una voce di accesso, Amazon EKS lo memorizza insieme alla voce di accesso.
- Ogni voce di accesso include un tipo. Il tipo di voce di accesso dipende dal tipo di risorsa a cui viene associata e non definisce le autorizzazioni. Se non specifichi un tipo, Amazon EKS lo imposta automaticamente su `STANDARD`
  - `EC2_LINUX`: per un ruolo IAM usato con nodi Linux o Bottlerocket autogestiti

  - `EC2_WINDOWS`: per un ruolo IAM usato con nodi autogestiti di Windows
  - `FARGATE_LINUX`- Per un ruolo IAM utilizzato con Fargate (Fargate AWS )
  - `HYBRID_LINUX`: per un ruolo IAM usato con nodi ibridi
  - `STANDARD`: tipo predefinito se non specificato
  - `EC2`: per le classi di nodi personalizzate EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Creazione di una voce di accesso alla classe di nodi".
  - Non è possibile modificare il tipo dopo aver creato la voce di accesso.
- Non è necessario creare una voce di accesso per un ruolo IAM utilizzato per un gruppo di nodi gestiti o un profilo Fargate. EKS creerà voci di accesso (se abilitate) o aggiornerà la mappa di configurazione dell'autenticazione (se non sono disponibili le voci di accesso)
- Se il tipo di voce di accesso è `STANDARD`, puoi specificare un nome utente per la voce di accesso. Se non specifichi un valore per il nome utente, Amazon EKS assegnerà automaticamente uno dei valori seguenti, in funzione del tipo di voce di accesso e del principale IAM da te indicato, che si tratti di un ruolo IAM o di un utente IAM. A meno che tu non abbia un motivo particolare per specificare il tuo nome utente, consigliamo di non specificarne uno e di lasciare che Amazon EKS lo generi automaticamente. Se decidi di specificare il tuo nome utente:
  - Tieni presente che non può iniziare con `system:`, `eks:`, `aws:`, `amazon:` o `iam:`.
  - Se il nome utente è per un ruolo IAM, ti consigliamo di aggiungere `{{SessionName}}` o `{{SessionNameRaw}}` alla fine del nome utente. Se aggiungi uno `{{SessionName}}` o `{{SessionNameRaw}}` al tuo nome utente, il nome utente deve includere i due punti prima di {{SessionName}}. Quando si assume questo ruolo, il nome del nome della sessione AWS STS specificato durante l'assunzione del ruolo viene passato automaticamente al cluster e viene visualizzato nei CloudTrail log. Ad esempio, non puoi avere un nome utente del tipo `john{{SessionName}}`. Il nome utente deve essere `:john{{SessionName}}` o `jo:hn{{SessionName}}`. Solo i due punti devono essere prima di `{{SessionName}}`. Il nome utente generato da Amazon EKS nella tabella seguente include un ARN. Poiché un ARN include i due punti, soddisfa questo requisito. I due punti non sono obbligatori se non includi `{{SessionName}}` nel nome utente. Nota che in `{{SessionName}}` il carattere speciale "@" viene sostituito con "-" nel nome della sessione. `{{SessionNameRaw}}` mantiene tutti i caratteri speciali nel nome della sessione.

| Il tipo di principale IAM | Tipo | Valore del nome utente che Amazon EKS imposta automaticamente |
|---|---|---|
| Utente | `STANDARD` | L'ARN dell'utente. Ad esempio: `arn:aws:iam::<111122223333>:user/<my-user>` |
| Ruolo | `STANDARD` | L'ARN STS del ruolo quando viene assunto. Amazon EKS aggiunge `{{SessionName}}` al ruolo.<br><br>Ad esempio: `arn:aws:sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}`<br><br>Se l'ARN del ruolo che hai specificato conteneva un percorso, Amazon EKS lo rimuove nel nome utente generato. |
| Ruolo | `EC2_LINUX` o `EC2_WINDOWS` | `system:node:{{EC2PrivateDNSName}}` |
| Ruolo | `FARGATE_LINUX` | `system:node:{{SessionName}}` |
| Ruolo | `HYBRID_LINUX` | `system:node:{{SessionName}}` |

Puoi modificare il nome utente dopo aver creato la voce di accesso.

- Se il tipo di una voce di accesso è `STANDARD` e desideri utilizzare l'autorizzazione RBAC di Kubernetes, puoi aggiungere uno o più nomi dei gruppi alla voce di accesso. Dopo aver creato una voce di accesso, puoi aggiungere e rimuovere i nomi dei gruppi. Affinché il principale IAM abbia accesso agli oggetti Kubernetes sul cluster, è necessario creare e gestire oggetti di autorizzazione basati sui ruoli (RBAC) Kubernetes. Creare oggetti Kubernetes `RoleBinding` o `ClusterRoleBinding` nel cluster che specificano il nome del gruppo come `subject` per `kind: Group`. Kubernetes autorizza l'accesso del principale IAM a qualsiasi oggetto del cluster specificato in un oggetto `Role` Kubernetes o `ClusterRole` specificato anche nel `roleRef` delle tue associazioni. Se specifichi nomi di gruppo, ti consigliamo di acquisire familiarità con gli oggetti di autorizzazione basata sui ruoli (RBAC) di Kubernetes. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

  > **Important**
  >
  > Amazon EKS non conferma che gli oggetti RBAC Kubernetes presenti nel cluster includano uno dei nomi di gruppo specificati. Ad esempio, se crei una voce di accesso per un gruppo che attualmente non esiste, Amazon EKS accetterà la configurazione senza restituire un errore, ma il responsabile IAM non avrà alcuna autorizzazione finché non verranno create le risorse RBAC Kubernetes corrispondenti.

  Puoi collegare policy di accesso Amazon EKS a una voce di accesso, come alternativa o complemento all'autorizzazione concessa da Kubernetes al principale IAM per accedere agli oggetti Kubernetes sul tuo cluster. Amazon EKS autorizza i principali IAM ad accedere agli oggetti Kubernetes sul tuo cluster con le autorizzazioni previste dalla policy di accesso. Puoi definire l'ambito delle autorizzazioni di una policy di accesso ai namespace di Kubernetes che specifichi. L'uso delle politiche di accesso non richiede la gestione degli oggetti RBAC di Kubernetes. Per ulteriori informazioni, consulta the section called "Associare le policy di accesso".
- Se crei una voce di accesso con tipo `EC2_LINUX` o `EC2_WINDOWS`, il principale IAM che crea la voce di accesso deve disporre dell'autorizzazione `iam:PassRole`. Per ulteriori informazioni, consulta Concessione a un utente delle autorizzazioni per passare un ruolo a un AWS servizio nella IAM User Guide.
- Analogamente al comportamento di IAM standard, la creazione e gli aggiornamenti delle voci di accesso alla fine sono coerenti e potrebbero essere necessari alcuni secondi prima che la chiamata API iniziale risulti completata con successo. È necessario progettare le applicazioni in modo da tenere in considerazione questi potenziali ritardi Si consiglia di non includere creazioni o

aggiornamenti degli accessi nei percorsi critici e ad alta disponibilità del codice dell'applicazione. Al contrario, apportare modifiche in un'inizializzazione separata o in una routine di configurazione che si esegue meno frequentemente. Inoltre, assicurarsi di verificare che le modifiche siano state propagate prima che i flussi di lavoro di produzione dipendano da esse.

- Le voci di accesso non supportano i ruoli collegati al servizio. Non puoi creare voci di accesso in cui l'ARN principale è un ruolo collegato al servizio. Puoi identificare i ruoli collegati al servizio in base al relativo ARN, che è nel formato `arn:aws: iam::*:role/aws-service-role/*`.

È possibile creare una voce di accesso utilizzando Console di gestione AWS o la CLI AWS .

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster in cui desideri creare una voce di accesso.
3. Scegliere la scheda Accesso.
4. Seleziona Crea voce di accesso.
5. Per Principale IAM, seleziona un ruolo o un utente IAM esistente. Le best practice IAM consigliano di accedere al cluster utilizzando ruoli IAM con credenziali a breve termine, anziché utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella IAM User Guide.
6. Per Tipo, se la voce di accesso è per il ruolo del nodo utilizzato per i nodi Amazon EC2 autogestiti, seleziona EC2 Linux o EC2 Windows. Altrimenti, accetta l'impostazione predefinita (Standard).
7. Se il Tipo che hai scelto è Standard e desideri specificare un Nome utente, inserisci il nome utente.
8. Se il Tipo che hai scelto è Standard e desideri utilizzare l'autorizzazione RBAC di Kubernetes per il principale IAM, specifica uno o più nomi per i Gruppi. Se non specifichi nomi dei gruppi e desideri utilizzare l'autorizzazione Amazon EKS, puoi associare una policy di accesso in un passaggio successivo o dopo la creazione della voce di accesso.
9. (Facoltativo) Per Tag, assegna etichette alla voce di accesso. Ad esempio, per facilitare la ricerca di tutte le risorse con lo stesso tag.
10. Scegli Avanti.
11. Nella pagina Aggiungi policy di accesso, se il tipo che hai scelto era Standard e desideri che Amazon EKS autorizzi il principale IAM a disporre delle autorizzazioni per gli oggetti Kubernetes sul tuo cluster, completa i seguenti passaggi. Altrimenti, scegli Next (Successivo).

a. Per Nome della policy, scegli una policy di accesso. Non puoi visualizzare le autorizzazioni delle policy di accesso, tuttavia includono autorizzazioni simili a quelle degli oggetti `ClusterRole` di Kubernetes rivolti all'utente. Per ulteriori informazioni, consulta User-facing i ruoli nella documentazione di Kubernetes.

b. Scegli una delle seguenti opzioni:

- Cluster: scegli questa opzione se desideri che Amazon EKS autorizzi il principale IAM a disporre delle autorizzazioni nella policy di accesso per tutti gli oggetti Kubernetes sul tuo cluster.
- Namespace Kubernetes: scegli questa opzione se desideri che Amazon EKS autorizzi il principale IAM a disporre delle autorizzazioni nella policy di accesso per tutti gli oggetti Kubernetes in uno specifico namespace Kubernetes sul tuo cluster. Per Namespace, inserisci il nome del namespace Kubernetes sul tuo cluster. Se desideri aggiungere altri spazi dei nomi, scegli Aggiungi nuovo spazio dei nomi e inserisci il nome dello spazio dei nomi.

c. Se desideri aggiungere ulteriori policy, scegli Aggiungi policy. Puoi definire l'ambito di ciascuna policy in modo diverso, ma puoi aggiungere ogni policy solo una volta.

d. Scegli Avanti.

12. Verifica la configurazione per la tua voce di accesso. Se noti qualcosa di errato, scegli Precedente per tornare indietro e correggere l'errore. Se la configurazione è corretta, scegli Crea.

AWS CLI

1. Installa la AWS CLI, come descritto nella Guida all'installazione nella AWS Command Line Interface User Guide.
2. Per creare una voce di accesso Per creare voci di accesso è possibile utilizzare uno qualsiasi degli esempi seguenti:
   - Crea una voce di accesso per un gruppo di nodi Amazon EC2 Linux autogestito. Sostituiscila *`my-cluster`* con il nome del tuo cluster, *`111122223333`* con l'ID AWS del tuo account e *`EKS-my-cluster-self-managed-ng-1`* con il nome del ruolo IAM del tuo nodo. Se il tuo gruppo di nodi è un gruppo di nodi Windows, sostituiscilo *`EC2_LINUX`* con`EC2_WINDOWS`.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_LINUX
```

Non puoi utilizzare l'opzione `--kubernetes-groups` quando specifichi un tipo diverso da `STANDARD`. Non puoi associare una policy di accesso a questa voce di accesso, perché il suo tipo è un valore diverso da `STANDARD`.

- Crea una voce di accesso che permetta a un ruolo IAM, non utilizzato per un gruppo di nodi autogestiti di Amazon EC2, di essere autorizzato da Kubernetes per accedere al tuo cluster. Sostituiscilo *`my-cluster`* con il nome del tuo cluster, *`111122223333`* con l'ID del tuo AWS account e *`my-role`* con il nome del tuo ruolo IAM. Sostituiscilo *`Viewers`* con il nome di un gruppo che hai specificato in un Kubernetes `RoleBinding` o in un `ClusterRoleBinding` oggetto sul tuo cluster.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role --type STANDARD --username Viewers --kubernetes-
groups Viewers
```

- Crea una voce di accesso che consenta a un utente IAM di autenticarsi nel tuo cluster. Questo esempio viene fornito perché è possibile, anche se le best practice IAM consigliano di accedere al cluster utilizzando ruoli IAM con credenziali a breve termine, anziché utenti IAM con credenziali a lungo termine. Per maggiori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella IAM User Guide.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:user/my-user --type STANDARD --username my-user
```

Se desideri che questo utente abbia un accesso al tuo cluster maggiore rispetto alle autorizzazioni nei ruoli di rilevamento delle API Kubernetes, è necessario associare una policy di accesso alla voce di accesso, poiché l'opzione `--kubernetes-groups` non viene utilizzata. Per ulteriori informazioni, consulta the section called "Associare le policy di accesso" e API discovery roles nella documentazione di Kubernetes.

## Aggiornamento delle voci di accesso

È possibile aggiornare una voce di accesso utilizzando Console di gestione AWS o AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.

2. Scegli il nome del cluster in cui desideri creare una voce di accesso.
3. Scegli la scheda Accesso.
4. Scegli la voce di accesso che desideri aggiornare.
5. Scegli Modifica.
6. Per Nome utente, puoi modificare il valore esistente.
7. Per Gruppi, puoi rimuovere i nomi dei gruppi esistenti o aggiungere nuovi nomi dei gruppi. Se esistono i seguenti nomi di gruppi, non rimuoverli: system:nodes o system:bootstrappers. La rimozione di questi gruppi può causare un malfunzionamento del cluster. Se non specifichi nomi dei gruppi e desideri utilizzare l'autorizzazione Amazon EKS, associa una policy di accesso in un passaggio successivo.
8. Per Tag, puoi assegnare etichette alla voce di accesso. Ad esempio, per facilitare la ricerca di tutte le risorse con lo stesso tag. Puoi anche rimuovere i tag esistenti.
9. Scegli Save changes (Salva modifiche).
10. Se desideri associare una policy di accesso alla voce, consulta la sezione the section called "Associare le policy di accesso".

CLI AWS

1. Installa AWS CLI come descritto in Installazione nella Guida per l'utente dell'interfaccia a riga di comando AWS.
2. Per aggiornare una voce di accesso, sostituisci *my-cluster* con il nome del cluster, *111122223333* con l'ID del tuo account AWS e *EKS-my-cluster-my-namespace-Viewers* con il nome di un ruolo IAM.

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-
groups Viewers
```

   Non è possibile utilizzare l'opzione `--kubernetes-groups` se il tipo di voce di accesso è un valore diverso da `STANDARD`. Inoltre, non è possibile associare una policy di accesso a una voce di accesso con un tipo diverso da `STANDARD`.

## Eliminare le voci di accesso

Se scopri di aver eliminato una voce di accesso per errore, puoi sempre ricrearla. Se la voce di accesso da eliminare è associata a policy di accesso, le associazioni vengono eliminate automaticamente. Non è necessario dissociare le policy di accesso da una voce di accesso prima di eliminarla.

È possibile eliminare una voce di accesso utilizzando la Console di gestione AWS o AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster da cui desideri eliminare una voce di accesso.
3. Scegli la scheda Accesso.
4. Nell'elenco Voci di accesso, seleziona la voce di accesso da eliminare.
5. Scegliere Delete (Elimina).
6. Nella finestra di dialogo di conferma, seleziona Elimina.

CLI AWS

1. Installare AWS CLI come descritto nella pagina Installing della Guida per l'utente dell'Interfaccia della linea di comando AWS.
2. Per eliminare una voce di accesso, sostituire *my-cluster* con il nome del cluster, *111122223333* con l'ID dell'account AWS e *my-role* con il nome del ruolo IAM a cui si desidera revocare l'accesso al cluster.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

## Impostazione di un nome utente personalizzato per le voci di accesso EKS

Quando crei voci di accesso per Amazon EKS, è possibile utilizzare il nome utente generato automaticamente o specificare un nome utente personalizzato. Questa pagina spiega entrambe le opzioni e ti guida nell'impostazione di un nome utente personalizzato.

Panoramica di

Il nome utente in una voce di accesso è utilizzato per identificare il principale IAM nei log e negli audit trail di Kubernetes. Per impostazione predefinita, Amazon EKS genera un nome utente basato sull'ARN dell'identità IAM, ma è possibile specificare un nome utente personalizzato se necessario.

Generazione di un nome utente predefinito

Se non specifichi un valore per il nome utente, Amazon EKS genera automaticamente un nome utente basato sull'identità IAM:

- Per un utente IAM:
  - EKS imposta il nome utente Kubernetes sull'ARN dell'utente IAM
  - Esempio:

```
{arn-aws}iam::<111122223333>:user/<my-user>
```

- Per ruoli IAM:
  - EKS imposta il nome utente Kubernetes in base all'ARN del ruolo IAM
  - L'ARN STS del ruolo quando è assunto. Amazon EKS aggiunge `{{SessionName}}` al ruolo. Se l'ARN del ruolo che hai specificato conteneva un percorso, Amazon EKS lo rimuove nel nome utente generato.
  - Esempio:

```
{arn-aws}sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}
```

A meno che tu non abbia un motivo particolare per specificare il tuo nome utente, consigliamo di non specificarne uno e di lasciare che Amazon EKS lo generi automaticamente.

Impostare un nome utente personalizzato

Quando si crea una voce di accesso, è possibile specificare un nome utente personalizzato utilizzando il parametro `--username`:

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD --username <custom-username>
```

Requisiti per nomi utente personalizzati

Se decidi specificare un nome utente personalizzato:

- Il nome utente non può iniziare con `system:`, `eks:`, `aws:`, `amazon:` o `iam:`.
- Se il nome utente è per un ruolo IAM, ti consigliamo di aggiungere `{{SessionName}}` o `{{SessionNameRaw}}` alla fine del nome utente.
  - Se aggiungi uno dei due `{{SessionName}}` o `{{SessionNameRaw}}` al tuo nome utente, il nome utente deve includere i due punti prima di {{SessionName}}.

## Crea una voce di accesso per un ruolo o un utente IAM utilizzando una policy di accesso e la AWS CLI

Crea voci di accesso Amazon EKS che utilizzano policy di accesso EKS AWS gestite per concedere alle identità IAM autorizzazioni standardizzate per l'accesso e la gestione dei cluster Kubernetes.

Panoramica di

Le voci di accesso in Amazon EKS definiscono il modo in cui le identità IAM (utenti e ruoli) possono accedere ai cluster Kubernetes e interagire con essi. Creando voci di accesso con le policy di accesso EKS, puoi:

- Concedere a utenti o a ruoli IAM specifici l'autorizzazione ad accedere al tuo cluster EKS
- Controlla le autorizzazioni utilizzando politiche di accesso EKS AWS gestite che forniscono set di autorizzazioni standardizzati e predefiniti
- Limitare le autorizzazioni a namespace specifici o a tutto il cluster
- Semplifica la gestione degli accessi senza modificare o creare risorse Kubernetes `aws-auth` ConfigMap RBAC
- Utilizza un approccio AWS integrato al controllo degli accessi a Kubernetes che copre i casi d'uso comuni mantenendo al contempo le migliori pratiche di sicurezza

Questo approccio è consigliato per la maggior parte dei casi d'uso perché fornisce autorizzazioni AWS gestite e standardizzate senza richiedere la configurazione manuale di Kubernetes RBAC. Le policy di accesso EKS eliminano la necessità di configurare manualmente le risorse Kubernetes RBAC e forniscono set di autorizzazioni predefiniti che coprono casi d'uso comuni.

Prerequisiti

- La modalità di autenticazione del cluster deve essere configurata per abilitare le voci di accesso. Per ulteriori informazioni, consulta the section called "Modalità di autenticazione".
- Installa e configura la AWS CLI, come descritto in Installazione nella Guida per l'utente dell'interfaccia a riga di AWS comando.

Fase 1: definire la voce di accesso

1. Trovare l'ARN dell'identità IAM, come un utente o un ruolo, a cui concedere le autorizzazioni.
   - Ogni identità IAM può disporre di una sola voce di accesso EKS.
2. Stabilisci se vuoi che le autorizzazioni della policy di accesso Amazon EKS si applichino solo a uno specifico namespace Kubernetes o all'intero cluster.
   - Se vuoi limitare le autorizzazioni a un namespace specifico, prendi nota del nome del namespace.
3. Seleziona la policy di accesso EKS che desideri per l'identità IAM. Questa policy fornisce le autorizzazioni all'interno del cluster. Annota l'ARN della policy.
   - Per un elenco delle policy, consulta le policy di accesso disponibili.
4. Stabilire se il nome utente generato automaticamente è adeguato per la voce di accesso o se è necessario specificarne uno manualmente.
   - AWS genera automaticamente questo valore in base all'identità IAM. È possibile impostare un nome utente personalizzato. Tale nome è visibile nei log di Kubernetes.
   - Per ulteriori informazioni, consulta the section called "Impostazione del nome utente personalizzato".

Fase 2: creare una voce di accesso

Dopo aver pianificato la voce di accesso, utilizza la AWS CLI per crearla.

L'esempio seguente copre la maggior parte dei casi d'uso. Visualizzare il riferimento CLI per tutte le opzioni di configurazione.

Nella fase successiva allegherai la policy di accesso.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD
```

Fase 3: associare la policy di accesso

Il comando varia a seconda che si voglia limitare la policy a un namespace Kubernetes specificato.

È necessario l'ARN della policy di accesso. Visualizza le policy di accesso disponibili.

Crea una policy senza l'ambito del namespace

```
aws eks associate-access-policy --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --access-scope type=cluster --policy-arn <access-policy-arn>
```

Crea con l'ambito del namespace

```
aws eks associate-access-policy --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 <access-policy-arn>
```

Fasi successive

- Create a kubeconfig so you can use kubectl with an IAM identity

## Creare una voce di accesso utilizzando i gruppi Kubernetes con AWS CLI

Creare voci di accesso Amazon EKS che utilizzano gruppi Kubernetes per l'autorizzazione e richiedono una configurazione di controllo degli accessi basato sul ruolo (RBAC) manuale.

> **Note**
>
> Per la maggior parte dei casi d'uso, consigliamo di utilizzare le policy di accesso EKS anziché l'approccio con gruppi Kubernetes descritto in questa pagina. Le policy di accesso EKS offrono un modo più semplice e integrato con AWS per gestire l'accesso senza richiedere la configurazione RBAC manuale. Utilizzare l'approccio con gruppi Kubernetes solo quando è necessario un controllo più granulare di quello offerto dalle policy di accesso EKS.

Panoramica

Le voci di accesso definiscono il modo in cui le identità IAM (utenti e ruoli) accedono ai cluster Kubernetes. L'approccio con gruppi Kubernetes concede agli utenti o ai ruoli IAM l'autorizzazione

ad accedere al cluster EKS tramite gruppi Kubernetes RBAC standard. Questo metodo richiede la creazione e la gestione di risorse Kubernetes RBAC (Roles, RoleBindings, ClusterRoles e ClusterRoleBindings) ed è consigliato quando sono necessari set di autorizzazioni altamente personalizzati, requisiti di autorizzazione complessi o quando si desidera mantenere modelli di controllo degli accessi coerenti in ambienti Kubernetes ibridi.

Questo argomento non tratta la creazione di voci di accesso per le identità IAM utilizzate dalle istanze Amazon EC2 per unirsi ai cluster EKS.

Prerequisiti

- La modalità di autenticazione del cluster deve essere configurata per abilitare le voci di accesso. Per ulteriori informazioni, consulta the section called "Modalità di autenticazione".
- Installare e configurare AWS CLI come descritto nella pagina Installing della Guida per l'utente dell'Interfaccia della linea di comando AWS.
- Si consiglia di avere familiarità con il controllo RBAC Kubernetes. Per ulteriori informazioni, consulta Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.

Fase 1: definire la voce di accesso

1. Trovare l'ARN dell'identità IAM, come un utente o un ruolo, a cui concedere le autorizzazioni.
   - Ogni identità IAM può disporre di una sola voce di accesso EKS.
2. Determinare quali gruppi Kubernetes devono essere associati a tale identità IAM.
   - Sarà necessario creare o utilizzare le risorse Kubernetes `Role`/`ClusterRole` e `RoleBinding`/`ClusterRoleBinding` esistenti che fanno riferimento a tali gruppi.
3. Stabilire se il nome utente generato automaticamente è adeguato per la voce di accesso o se è necessario specificarne uno manualmente.
   - AWS genera questo valore automaticamente in base all'identità IAM. È possibile impostare un nome utente personalizzato. Tale nome è visibile nei log di Kubernetes.
   - Per ulteriori informazioni, consulta the section called "Impostazione del nome utente personalizzato".

Fase 2: creare una voce di accesso con i gruppi Kubernetes

Dopo aver definito la voce di accesso, utilizzare AWS CLI per crearla con i gruppi Kubernetes adeguati.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD --kubernetes-groups <groups>
```

Sostituire:

- `<cluster-name>` con il nome del cluster EKS.
- `<iam-identity-arn>` con l'ARN del ruolo o dell'utente IAM.
- `<groups>` con un elenco separato da virgole di gruppi Kubernetes (ad esempio, "system:developers,system:readers").

Visualizzare il riferimento CLI per tutte le opzioni di configurazione.

Fase 3: configurazione del controllo RBAC Kubernetes

Affinché il principale IAM abbia accesso agli oggetti Kubernetes nel cluster, è necessario creare e gestire oggetti del controllo degli accessi basato sul ruolo (RBAC) Kubernetes:

1. Creare oggetti Kubernetes `Role` o `ClusterRole` che definiscono le autorizzazioni.
2. Creare oggetti Kubernetes `RoleBinding` o `ClusterRoleBinding` nel cluster che specificano il nome del gruppo come `subject` per `kind: Group`.

Per informazioni dettagliate sulla configurazione di gruppi e autorizzazioni in Kubernetes, consultare Using RBAC Authorization nella documentazione di Kubernetes.

Passaggi successivi

- Create a kubeconfig so you can use kubectl with an IAM identity

## Concedi agli utenti IAM l'accesso a Kubernetes con un ConfigMap

> **Important**
>
> La funzionalità `aws-auth ConfigMap` è obsoleta. Per il metodo consigliato per gestire l'accesso alle API Kubernetes, consultare the section called "Voci di accesso".

L'accesso al cluster utilizzando i principali IAM è abilitato dall'autenticatore AWS IAM per Kubernetes, che viene eseguito sul piano di controllo di Amazon EKS. L'autenticatore riceve le informazioni di configurazione da `aws-auth ConfigMap`. Per tutte le impostazioni di `ConfigMap aws-auth`, consulta Formato di configurazione completo su GitHub.

## Aggiunta di principali IAM al cluster Amazon EKS

Quando crei un cluster Amazon EKS, il principale IAM che crea il cluster riceve automaticamente le autorizzazioni `system:masters` nella configurazione del controllo degli accessi basato sul ruolo (RBAC) nel piano di controllo di Amazon EKS. Questo principale non è visualizzato in una configurazione visibile qualsiasi, quindi assicurati di tenere traccia di quale principale ha originariamente creato il cluster. Per concedere a ulteriori principali IAM la capacità di interagire con il cluster, modifica `aws-auth ConfigMap` all'interno di Kubernetes e crea `rolebinding` o `clusterrolebinding` Kubernetes con il nome di `group` specificato in `aws-auth ConfigMap`.

> **Note**
>
> Per ulteriori informazioni sulla configurazione del controllo degli accessi basato sul ruolo (RBAC) di Kubernetes, consultare Using RBAC Authorization nella documentazione di Kubernetes.

1. Determina quali credenziali `kubectl` vengono utilizzate per accedere al cluster. Sul computer, è possibile vedere quali credenziali `kubectl` utilizza con il seguente comando. Sostituiscilo *`~/.kube/config`* con il percorso del tuo `kubeconfig` file se non usi il percorso predefinito.

   ```
   cat ~/.kube/config
   ```

   Di seguito viene riportato un output di esempio.

   ```
   [...]
   contexts:
   - context:
       cluster: my-cluster.region-code.eksctl.io
       user: admin@my-cluster.region-code.eksctl.io
     name: admin@my-cluster.region-code.eksctl.io
   current-context: admin@my-cluster.region-code.eksctl.io
   [...]
   ```

Nell'output dell'esempio precedente, le credenziali per un utente denominato *admin* sono configurate per un cluster denominato *my-cluster*. Se si tratta dell'utente che ha creato il cluster, ha già accesso al cluster. Se l'utente non ha creato il cluster, devi completare i passaggi rimanenti per abilitare l'accesso al cluster per gli altri principali IAM. Le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti. Per visualizzare gli altri principali che hanno attualmente accesso al cluster, esegui il comando seguente:

```
kubectl describe -n kube-system configmap/aws-auth
```

Di seguito viene riportato un output di esempio.

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws: iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
====

Events:  <none>
```

L'esempio precedente è una `aws-auth` `ConfigMap` predefinita. Solo il ruolo dell'istanza del nodo ha accesso al cluster.

2. Assicurati di disporre di `roles` e `rolebindings` o `clusterroles` e `clusterrolebindings` di Kubernetes da mappare ai principali IAM. Per ulteriori informazioni su queste risorse, consulta Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.
   a. Visualizza i `roles` o `clusterroles` Kubernetes esistenti. L'ambito di `Roles` è `namespace`, mentre l'ambito di `clusterroles` è il cluster.

```
kubectl get roles -A
```

```
kubectl get clusterroles
```

b. Visualizza i dettagli dei valori `role` o `clusterrole` restituiti nell'output precedente e verifica di disporre delle autorizzazioni (`rules`) da concedere ai principali IAM del cluster.

   Sostituisci *`role-name`* con un nome del `role` restituito nell'output del comando precedente. Sostituisci *`kube-system`* con lo spazio dei nomi del `role`.

```
kubectl describe role role-name -n kube-system
```

   Sostituisci *`cluster-role-name`* con un nome del `clusterrole` restituito nell'output del comando precedente.

```
kubectl describe clusterrole cluster-role-name
```

c. Visualizza i `rolebindings` o `clusterrolebindings` Kubernetes esistenti. L'ambito di `Rolebindings` è `namespace`, mentre l'ambito di `clusterrolebindings` è il cluster.

```
kubectl get rolebindings -A
```

```
kubectl get clusterrolebindings
```

d. Visualizza i dettagli di `rolebinding` o `clusterrolebinding` e verifica che dispongano di un `role` or `clusterrole` della fase precedente elencato come un `roleRef` e un nome di gruppo elencato per `subjects`.

   Sostituisci *`role-binding-name`* con un nome del `rolebinding` restituito nell'output del comando precedente. Sostituisci *`kube-system`* con lo `namespace` del `rolebinding`.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

   Di seguito viene riportato un output di esempio.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
```

```
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

Sostituisci *cluster-role-binding-name* con un nome del `clusterrolebinding` restituito nell'output del comando precedente.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

Di seguito viene riportato un output di esempio.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. Modificare `aws-auth` `ConfigMap`. Per aggiornare `ConfigMap` puoi usare uno strumento adeguato, ad esempio `eksctl`, oppure puoi eseguire l'aggiornamento in modo manuale tramite modifica.

> **Important**
>
> Ti consigliamo di utilizzare `eksctl`, o uno strumento simile, per modificare `ConfigMap`. Per informazioni su altri strumenti che puoi utilizzare, consulta Utilizzare gli strumenti per apportare modifiche a aws-auth ConfigMap nelle guide alle best practice di Amazon

EKS. Una formattazione impropria di `aws-auth` `ConfigMap` può causare la perdita dell'accesso al cluster.

- Visualizzare le fasi per modificare configmap con eksctl.
- Visualizzare le fasi per modificare configmap manualmente.

### Modifica con eksctl ConfigMap

1. È necessaria la versione `0.215.0` o successiva dello strumento a riga di `eksctl` comando installata sul dispositivo oppure. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Visualizza le mappature correnti in `ConfigMap`. Sostituisci *`my-cluster`* con il nome del cluster. Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster.

   ```
   eksctl get iamidentitymapping --cluster my-cluster --region=region-code
   ```

   Di seguito viene riportato un output di esempio.

   ```
   ARN
               USERNAME                                GROUPS
    ACCOUNT
   arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
   NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
    system:bootstrappers,system:nodes
   ```

3. Aggiungi una mappatura per un ruolo. Sostituisci *`my-role`* con il nome del tuo ruolo. Sostituisci *`eks-console-dashboard-full-access-group`* con il nome del gruppo specificato nel tuo Kubernetes `RoleBinding` o nell'oggetto. `ClusterRoleBinding` Sostituisci *`111122223333`* con l'ID del tuo account. Puoi sostituire *`admin`* con un nome a tua scelta.

   ```
   eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
       --arn arn:aws: iam::111122223333:role/my-role --username admin --group eks-
   console-dashboard-full-access-group \
       --no-duplicate-arns
   ```

> **Important**
>
> L'ARN del ruolo non può includere un percorso, ad esempio `role/my-team/developers/my-role`. Il formato dell'ARN deve essere `arn:aws:iam::111122223333:role/my-role` . In questo esempio, `my-team/developers/` deve essere rimosso.

Di seguito viene riportato un output di esempio.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "{arn-aws}iam::111122223333:role/my-role" to
 auth ConfigMap
```

4. Aggiungi una mappatura per un utente. Le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti. Sostituisci *my-user* con il tuo nome utente. Sostituisci *eks-console-dashboard-restricted-access-group* con il nome del gruppo specificato nel tuo `RoleBinding` Kubernetes o nell'oggetto. `ClusterRoleBinding` Sostituisci *111122223333* con l'ID del tuo account. Puoi sostituire *my-user* con un nome a tua scelta.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws: iam::111122223333:user/my-user --username my-user --group eks-
console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

Di seguito viene riportato un output di esempio.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws: iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. Visualizza nuovamente le mappature nella `ConfigMap`.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Di seguito viene riportato un output di esempio.

```
ARN
           USERNAME                                GROUPS
      ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws: iam::111122223333:role/admin
              my-role                                   eks-console-dashboard-full-
access-group
arn:aws: iam::111122223333:user/my-user
            my-user                                  eks-console-dashboard-restricted-
access-group
```

Modifica manualmente ConfigMap

1. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> Se ricevi un errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`«, utilizza la procedura descritta in Applica aws-auth ConfigMap al tuo cluster per applicare lo stock `ConfigMap`

2. Aggiungi i tuoi principali IAM alla `ConfigMap`. Un gruppo IAM non è un principale IAM, quindi non può essere aggiunto alla `ConfigMap`.
   - Per aggiungere un ruolo IAM (ad esempio, per utenti federati): aggiungere i dettagli del ruolo alla sezione `mapRoles` della `ConfigMap`, in `data`. Aggiungi questa sezione se non esiste già nel file. Ogni voce supporta i seguenti parametri:
     - rolearn: l'ARN; del ruolo IAM da aggiungere. Questo valore non può includere un percorso. Ad esempio, non puoi specificare un ARN, come `arn:aws: iam::111122223333:role/my-team/developers/role-name` . L'ARN deve essere `arn:aws: iam::111122223333:role/role-name` .
     - username: il nome utente all'interno di Kubernetes da associare al ruolo IAM.

- groups: il gruppo o l'elenco dei gruppi Kubernetes a cui mappare il ruolo. Tale gruppo può essere un gruppo predefinito o un gruppo specificato in un `clusterrolebinding` o un `rolebinding`. Per ulteriori informazioni consultare Default roles and role bindings nella documentazione di Kubernetes.

- Per aggiungere un utente IAM: le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti. Aggiungi i dettagli dell'utente alla sezione `mapUsers` della `ConfigMap`, in `data`. Aggiungi questa sezione se non esiste già nel file. Ogni voce supporta i seguenti parametri:
  - userarn: L'ARN; dell'utente IAM da aggiungere.
  - username: Il nome utente all'interno di Kubernetes da associare all'utente IAM.
  - groups: il gruppo o l'elenco dei gruppi Kubernetes a cui mappare l'utente. Tale gruppo può essere un gruppo predefinito o un gruppo specificato in un `clusterrolebinding` o un `rolebinding`. Per ulteriori informazioni consultare Default roles and role bindings nella documentazione di Kubernetes.

=== Variabili modello per e `username` `groups`

I `groups` campi `username` and in entrambe le `mapRoles` `mapUsers` voci supportano le variabili di modello che AWS IAM Authenticator sostituisce al momento dell'autenticazione. Sono disponibili le seguenti variabili di modello:

`{{AccountID}}`

L'ID dell' AWS account a 12 cifre del principale IAM autenticato.

`{{SessionName}}`

Il nome della sessione del ruolo STS, con `@` caratteri sostituiti da. `-` Per i ruoli delle istanze EC2, questo è l'ID dell'istanza. Per i ruoli federati, questa è l'identità federata. Per i ruoli assunti `consts:AssumeRole`, questo è il valore del `RoleSessionName` parametro impostato dal chiamante.

`{{SessionNameRaw}}`

Il nome grezzo della sessione del ruolo STS senza alcuna sostituzione di caratteri.

`{{AccessKeyID}}`

L'ID della chiave di AWS accesso utilizzato per autenticare la richiesta.

`{{EC2PrivateDNSName}}`

Il nome DNS privato dell'istanza EC2. Questa variabile richiede che il nome della sessione sia un ID di istanza EC2 valido ed effettua una chiamata `ec2:DescribeInstances` API.

> **Important**
>
> I `{{SessionNameRaw}}` valori `{{SessionName}}` e sono controllati dall'utente quando si assume un ruolo con. `sts:AssumeRole` Il chiamante imposta il `RoleSessionName` parametro su una stringa quasi arbitraria. Se utilizzi queste variabili nel `username` campo, un chiamante che può assumere il ruolo mappato può impersonare qualsiasi nome utente Kubernetes che non corrisponde a un prefisso riservato (come,,, o). `system: eks: aws: amazon: iam:` Se utilizzi queste variabili nel `groups` campo, un chiamante può scegliere la propria appartenenza al gruppo Kubernetes creando il nome della sessione. Usa o `{{SessionNameRaw}}` in `{{SessionName}}` `username` o `groups` se ti fidi di tutti i presidi che possono assumere il ruolo.

Ad esempio, il blocco YAML seguente contiene:

- Una sezione `mapRoles` che mappa l'istanza del nodo IAM ai gruppi Kubernetes, in modo che i nodi possano registrarsi autonomamente con il cluster e il ruolo IAM `my-console-viewer-role` mappato a un gruppo Kubernetes in grado di visualizzare tutte le risorse Kubernetes per tutti i cluster. Per un elenco delle autorizzazioni del gruppo IAM e Kubernetes necessarie per il ruolo IAM `my-console-viewer-role`, consulta the section called "Autorizzazioni richieste".
- Una `mapUsers` sezione che mappa l'utente `admin` IAM dall' AWS account predefinito al gruppo `system:masters` Kubernetes e l'`my-user`utente di un AWS account diverso mappato a un gruppo Kubernetes in grado di visualizzare le risorse Kubernetes per uno spazio di nomi specifico. Per un elenco delle autorizzazioni del gruppo IAM e Kubernetes necessarie per l'utente IAM `my-user`, consulta the section called "Autorizzazioni richieste".

Aggiungi o rimuovi righe in base alle esigenze e sostituisci tutti i valori di esempio con quelli desiderati.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will
 be
```

```
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws: iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws: iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws: iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws: iam::444455556666:user/my-user
      username: my-user
```

1. Salva il file ed esci dall'editor di testo.

## Applica aws-auth al tuo cluster **ConfigMap**

La `aws-auth` `ConfigMap` viene creata e applicata automaticamente al cluster quando si crea un gruppo di nodi gestito o quando si crea un gruppo di nodi utilizzando `eksctl`. È inizialmente creata per consentire ai nodi di unirsi al cluster, ma questa `ConfigMap` può essere usata anche per aggiungere l'accesso RBAC (controllo degli accessi basato sul ruolo) a principali IAM. Se non hai avviato nodi autogestiti e non hai applicato la `aws-auth` `ConfigMap` al cluster, puoi farlo utilizzando la procedura seguente.

1. Controlla per vedere se è già stata applicata la `aws-auth` `ConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

Se ricevi un errore di tipo "`Error from server (NotFound): configmaps "aws-auth" not found`", procedi con le fasi seguenti per applicare lo stock `ConfigMap`.

2. Scarica, modifica e applica la mappa di configurazione dell' AWS autenticatore.

   a. Scarica la mappa di configurazione.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. Nel file `aws-auth-cm.yaml`, imposta il `rolearn` sul nome della risorsa Amazon (ARN) del ruolo IAM associato ai nodi. Per eseguire questa operazione, utilizza un editor di testo o sostituisci *`my-node-instance-role`* eseguendo il comando seguente:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
   ```

   Non modificare altre righe in questo file.

   > **Important**
   >
   > L'ARN del ruolo non può includere un percorso, ad esempio `role/my-team/developers/my-role`. Il formato dell'ARN deve essere `arn:aws:iam::`*`111122223333`*`:role/`*`my-role`* . In questo esempio, `my-team/developers/` deve essere rimosso.

   Puoi controllare gli output dello AWS CloudFormation stack per i tuoi gruppi di nodi e cercare i seguenti valori:

   - InstanceRoleARN: per i gruppi di nodi creati con `eksctl`
   - NodeInstanceRole— Per i gruppi di nodi creati con i AWS CloudFormation modelli forniti da Amazon EKS nel Console di gestione AWS

   c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso non autorizzato o negato (kubectl) " nell'argomento relativo alla risoluzione dei problemi.

3. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

Inserisci `Ctrl+C` per tornare a un prompt della shell.

## Concedere agli utenti l'accesso a Kubernetes con un provider OIDC esterno

Amazon EKS supporta l'utilizzo di provider di identità OpenID Connect (OIDC) come metodo per autenticare gli utenti nel cluster. I provider di identità OIDC possono essere utilizzati con o in alternativa all' AWS Identity and Access Management (IAM). Per ulteriori informazioni sull'utilizzo di IAM, consultare the section called "Accesso API Kubernetes". Dopo aver configurato l'autenticazione per il cluster, è possibile creare `roles` e `clusterroles` Kubernetes per assegnare autorizzazioni ai ruoli e quindi associare i ruoli alle identità utilizzando `rolebindings` e `clusterrolebindings` Kubernetes. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

- Kubernetes non fornisce un provider di identità OIDC. È possibile utilizzare un provider di identità OIDC pubblico esistente oppure eseguire il proprio provider di identità. Per un elenco dei provider certificati, consultare OpenID Certification sul sito OpenID.
- L'URL dell'emittente del provider di identità OIDC deve essere accessibile pubblicamente, in modo che Amazon EKS possa individuare le chiavi di firma. Amazon EKS non supporta i provider di identità OIDC con certificati autofirmati.
- Puoi disabilitare l'autenticazione IAM nel cluster, perché sarà ancora necessaria per aggiungere i nodi a un cluster.
- Diversamente da un utente di provider di identità, un cluster Amazon EKS deve ancora essere creato da un principale IAM AWS . Questo perché il creatore del cluster interagisce con le API Amazon EKS, piuttosto che con le API Kubernetes.

- Gli utenti autenticati dal provider di identità OIDC sono elencati nel registro di controllo del cluster se CloudWatch i log sono attivati per il piano di controllo. Per ulteriori informazioni, consulta the section called "Abilitare o disabilitare i log del piano di controllo (control-plane)".
- Non puoi accedere Console di gestione AWS con un account di un provider OIDC. Puoi accedere solo Console di gestione AWS con un account the section called "Accesso alle risorse di cluster" di gestione delle AWS identità e degli accessi.

Il numero di provider di identità OIDC che puoi associare a un cluster dipende dalla versione Kubernetes del cluster. Sui cluster che eseguono Kubernetes versione 1.32 o successiva, puoi associare fino a 10 provider di identità OIDC. La dimensione combinata di tutte le configurazioni dei provider OIDC deve essere inferiore a 12 KB. Nei cluster che eseguono una versione precedente alla 1.32, è possibile associare un solo provider di identità OIDC.

- L'URL (`issuerUrl`) e il nome (`name`) dell'emittente devono essere univoci per tutti i provider di identità OIDC associati al cluster. Non puoi associare lo stesso provider più di una volta.
- L'associazione di un provider di identità OIDC è un aggiornamento del cluster. Il cluster entra nello `UPDATING` stato e la modifica può richiedere diversi minuti per essere applicata completamente ai server API del cluster. È possibile tenere traccia dello stato di avanzamento dell'aggiornamento con l'DescribeUpdateoperazione.

> **Note**
>
> Il piano di controllo recupera le chiavi di firma del provider dall'URL dell'emittente, pertanto l'emittente deve essere raggiungibile dal piano di controllo. Se il cluster utilizza il piano di controllo egress (`controlPlaneEgressMode=CUSTOMER_ROUTED`) indirizzato dal cliente, assicurati che l'endpoint dell'emittente sia raggiungibile tramite il percorso di uscita configurato nel tuo VPC. Per ulteriori informazioni, consulta the section called "Uscita del piano di controllo".

## Associazione di un provider di identità OIDC

Prima di associare un provider di identità OIDC al cluster, è necessario ottenere dal provider le seguenti informazioni:

URL dell'emittente

L'URL del provider di identità OIDC che consente al server API di individuare le chiavi di firma pubbliche per la verifica dei token. L'URL deve iniziare con `https://` e deve corrispondere alla richiesta `iss` nei token ID OIDC del provider. In conformità con lo standard OIDC, i componenti del percorso sono consentiti, ma i parametri di query non lo sono. In genere l'URL è costituito solo da un nome host, come `https://server.example.org` o `https://example.com`. Questo URL dovrebbe puntare al livello sottostante `.well-known/openid-configuration` e dovrà essere accessibile pubblicamente tramite Internet. L'URL dell'emittente deve essere univoco per tutti i provider di identità OIDC associati al cluster.

ID client (noto anche come pubblico)

L'ID per l'applicazione client che effettua richieste di autenticazione al provider di identità OIDC.

È possibile associare un provider di identità utilizzando `eksctl` o il Console di gestione AWS.

Associazione di un provider di identità usando eksctl

1. Crea un file denominato `associate-identity-provider.yaml` con i seguenti contenuti. Sostituire i valori di esempio con i propri valori. I valori nella sezione `identityProviders` vengono ottenuti dal provider di identità OIDC. I valori sono obbligatori solo per il `name`, `type`, `issuerUrl`, e `clientId` impostazioni in `identityProviders`.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
```

```
    requiredClaims:
      string: string
    tags:
      env: dev
```

> **Important**
>
> Non specificare `system:`, o qualsiasi parte di quella stringa, per `groupsPrefix` o `usernamePrefix`.

2. Creare il provider.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. Per utilizzare `kubectl` per lavorare con il cluster e il provider di identità OIDC, consulta Using kubectl nella documentazione di Kubernetes.

Associa un provider di identità utilizzando il AWS Console

1. Aprire la Console Amazon EKS.
2. Seleziona il cluster, poi seleziona la scheda Accesso.
3. Nella sezione Provider di identità OIDC, seleziona Associate Identity Provider.
4. Nella pagina Associate provider di identità OIDC, inserire o selezionare le seguenti opzioni e quindi selezionare Associa.
   - In Nome, inserire un nome univoco per il provider.
   - Per URL dell'emittente, inserire l'URL per il provider. Questo URL deve essere accessibile in Internet.
   - Per ID client, inserire l'ID client del provider di identità OIDC (noto anche come audience).
   - Per Richiesta Nome Utente, inserire la richiesta da utilizzare come nome utente.
   - Per Richiesta di Gruppi, inserire la richiesta da utilizzare come gruppo dell'utente.
   - (Opzionale) Selezionare Opzioni avanzate, inserire o selezionare le seguenti informazioni.
     - Prefisso Nome utente: inserire un prefisso da anteporre alle richieste del nome utente. Il prefisso viene anteposto alle richieste di nome utente per evitare conflitti con i nomi esistenti. Se non si fornisce un valore e il nome utente è un valore diverso da `email`, il prefisso viene impostato per impostazione predefinita sul valore per URL dell'emittente. Puoi utilizzare il

valore - per disabilitare tutti i prefissi. Non specificare `system:` o qualsiasi parte di quella stringa.

- Prefisso Gruppi: inserire un prefisso da anteporre alle richieste di gruppo. Il prefisso viene anteposto alle attestazioni di gruppo per evitare conflitti con nomi esistenti (ad esempio). `system:groups` Ad esempio, il valore `oidc:` crea nomi di gruppo come `oidc:engineering` e `oidc:infra`. Non specificare `system:` o qualsiasi parte di quella stringa.
- Richieste necessarie: selezionare Aggiungi richiesta e inserire una o più coppie di valori chiave che descrivono le richieste necessarie nel token ID client. Le coppie descrivono le richieste necessarie nel Token ID. Se impostato, ogni richiesta viene verificata per essere presente nel token ID con un valore corrispondente.
  a. Per utilizzare `kubectl` per lavorare con il cluster e il provider di identità OIDC, consulta Using kubectl nella documentazione di Kubernetes.

## Procedure consigliate per più provider di identità

Usa nome utente e prefissi di gruppo: imposta un prefisso nome utente (`usernamePrefix`) e un prefisso di gruppo (`groupsPrefix`) per ogni provider di identità OIDC per ridurre la possibilità che le identità di diversi provider si sovrappongano. Se non utilizzi i prefissi, utilizza le dichiarazioni relative al nome utente e ai gruppi che sono uniche per ogni provider. Entrambi gli approcci assicurano che due provider non possano produrre lo stesso nome utente o gruppo Kubernetes.

Convalida le dichiarazioni dei token: utilizza required claims (`requiredClaims`) per specificare le coppie chiave-valore che devono essere presenti nel token ID, in modo che il cluster accetti solo i token che autorizzi.

Limita le attestazioni in ogni token: configura il tuo provider di identità in modo che emetta token solo con le attestazioni necessarie al tuo cluster. La limitazione delle attestazioni riduce le informazioni esposte se un token viene compromesso.

## Policy IAM di esempio

Se si desidera impedire che un provider di identità OIDC venga associato a un cluster, creare e associare le seguenti policy IAM agli account IAM degli amministratori Amazon EKS. Per ulteriori informazioni, consulta Creating IAM policies e Adding IAM identity permissions nella Guida per l'utente IAM e nelle Actions in Service Authorization Reference.

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

La seguente policy di esempio consente l'associazione del provider di identità OIDC se il `clientID` è `kubernetes` e la `issuerUrl` è `https://cognito-idp.us-west-2.amazonaws.com/*`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
```

```
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

## Dissociazione di un provider di identità OIDC dal cluster

Se si dissocia un provider di identità OIDC dal cluster, gli utenti inclusi nel provider non potranno più accedere al cluster. Tuttavia, è comunque possibile accedere al cluster con principali IAM.

La dissociazione di un provider di identità OIDC è un aggiornamento del cluster. Il cluster entra nello `UPDATING` stato e la modifica può richiedere diversi minuti per essere applicata completamente ai server API del cluster. È possibile tenere traccia dello stato di avanzamento dell'aggiornamento con l'DescribeUpdateoperazione. Se al cluster sono associati più provider di identità OIDC, è necessario dissociarli uno alla volta. Attendi il completamento di ogni aggiornamento del cluster prima di dissociare il provider successivo.

1. Aprire la Console Amazon EKS.
2. Nella sezione Provider di identità OIDC, selezionare Dissocia, inserire il nome del provider di identità e quindi selezionare `Disassociate`.

# Visualizza le risorse Kubernetes nel Console di gestione AWS

Puoi visualizzare le risorse Kubernetes implementate nel cluster utilizzando la Console di gestione AWS, Non puoi visualizzare le risorse Kubernetes con la CLI o AWS eksctl. Per visualizzare le risorse Kubernetes con uno strumento a riga di comando, utilizza kubectl.

> **Note**
>
> Per visualizzare la scheda Risorse e la sezione Nodi nella scheda Compute Console di gestione AWS, il principale IAM che stai utilizzando deve disporre di autorizzazioni IAM e Kubernetes specifiche. Per ulteriori informazioni, consulta the section called "Autorizzazioni richieste".

1. Aprire la Console Amazon EKS.
2. Nell'elenco Cluster, seleziona il cluster contenente le risorse Kubernetes da visualizzare.
3. Selezionare la scheda Risorse.
4. Seleziona un gruppo Resource type (Tipi di risorse) per il quale desideri visualizzare le risorse, ad esempio Workloads (Carichi di lavoro). Viene visualizzato un elenco dei tipi di risorse in tale gruppo.
5. Seleziona un tipo di risorsa, ad esempio Deployments (Implementazioni), nel gruppo Workloads (Carichi di lavoro). Viene visualizzata una descrizione del tipo di risorsa, un collegamento alla documentazione Kubernetes per ulteriori informazioni sul tipo di risorsa e un elenco di risorse simili implementate nel cluster. Se l'elenco è vuoto, nel cluster non sono presenti risorse appartenenti a quel tipo specifico.
6. Per visualizzare ulteriori informazioni su una risorsa, selezionarla. Prova gli esempi seguenti:
   - Seleziona il gruppo Workloads (Carichi di lavoro), scegli il tipo di risorsa Deployments (Implementazioni), quindi seleziona la risorsa coredns. Per impostazione predefinita, la selezione di una risorsa avviene in una Visualizzazione strutturata. Per alcuni tipi di risorse, nella Visualizzazione strutturata è presente una sezione Pod, Questa sezione elenca i pod gestiti dal carico di lavoro. È possibile selezionare qualsiasi pod presente in elenco per visualizzarne le informazioni. Non tutti i tipi di risorse visualizzano le informazioni nella Visualizzazione strutturata. Selezionando Raw view (Visualizzazione non elaborata) nell'angolo in alto a destra della pagina della risorsa, viene visualizzata la risposta JSON completa dall'API Kubernetes per la risorsa.
   - Seleziona il gruppo Cluster, quindi il tipo di risorsa Nodes (Nodi). Viene visualizzato un elenco di tutti i nodi del cluster. I nodi possono appartenere a qualsiasi tipo di nodo di Amazon EKS. Si tratta dello stesso elenco visualizzato nella sezione Nodes (Nodi) quando si seleziona la scheda Compute (Calcolo) del cluster. Seleziona una risorsa di nodo dall'elenco. Nella Vista strutturata è presente anche la sezione Pod Questa sezione mostra tutti i pod in esecuzione sul nodo.

# Autorizzazioni richieste

Per visualizzare la scheda Risorse e la sezione Nodi nella scheda Compute Console di gestione AWS, il principale IAM che stai utilizzando deve disporre di autorizzazioni IAM e Kubernetes minime specifiche. È necessario che le autorizzazioni IAM e Kubernetes RBAC siano configurate correttamente. Completa i passaggi seguenti per assegnare le autorizzazioni richieste ai principali IAM.

1. Assicurati che `eks:AccessKubernetesApi` e le altre autorizzazioni IAM necessarie per visualizzare le risorse Kubernetes siano assegnati al principale IAM che stai utilizzando. Per ulteriori informazioni su come modificare le autorizzazioni per un principale IAM, consulta Controllo dell'accesso per i principali IAM nella Guida per l'utente di IAM. Per ulteriori informazioni su come modificare le autorizzazioni per un ruolo, consulta Modifica di una policy di autorizzazioni del ruolo (console) nella Guida per l'utente IAM.

   La seguente policy esemplificativa include le autorizzazioni necessarie affinché un principale possa visualizzare le risorse Kubernetes per tutti i cluster dell'account. Sostituiscilo con l'ID del tuo account. *111122223333* AWS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:ListFargateProfiles",
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:ListUpdates",
                "eks:AccessKubernetesApi",
                "eks:ListAddons",
                "eks:DescribeCluster",
                "eks:DescribeAddonVersions",
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
        }
    ]
}
```

Per visualizzare i nodi in cluster connessi, il ruolo IAM del connettore Amazon EKS dovrebbe essere in grado di rappresentare il principale nel cluster. Ciò consente ad Amazon EKS Connector di mappare il principale IAM a un utilizzatore Kubernetes.

2. Configura le autorizzazioni RBAC di Kubernetes utilizzando le voci di accesso EKS.

   Cosa sono gli EKS Access Entries?

   Le voci di accesso EKS sono un modo semplificato per concedere ai principali IAM (utenti e ruoli) l'accesso al tuo cluster Kubernetes. Invece di gestire manualmente le risorse RBAC di Kubernetes e le `aws-auth` ConfigMap, le voci di accesso gestiscono automaticamente la mappatura tra le autorizzazioni IAM e Kubernetes utilizzando le politiche gestite fornite da. AWS Per informazioni dettagliate the section called "Voci di accesso" sulle voci di accesso, consulta. Per informazioni sui criteri di accesso disponibili e sulle relative autorizzazioni, vedere Autorizzazioni dei criteri di accesso.

   Puoi allegare le autorizzazioni Kubernetes per accedere alle voci in due modi:

   - Usa una politica di accesso: le politiche di accesso sono modelli di autorizzazioni Kubernetes predefiniti gestiti da. AWS Questi forniscono set di autorizzazioni standardizzati per casi d'uso comuni.
   - Fai riferimento a un gruppo Kubernetes: se associ un'identità IAM a un gruppo Kubernetes, puoi creare risorse Kubernetes che concedono le autorizzazioni al gruppo. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

     a. Crea una voce di accesso per il tuo principale IAM utilizzando la AWS CLI. Sostituisci *my-cluster* con il nome del cluster. Sostituisci *111122223333* con l'ID del tuo account.

     ```
     aws eks create-access-entry \
         --cluster-name my-cluster \
         --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
     ```

     Di seguito viene riportato un output di esempio.

     ```
     {
     ```

```
    "accessEntry": {
        "clusterName": "my-cluster",
        "principalArn": "arn:aws: iam::111122223333:role/my-console-viewer-
role",
        "kubernetesGroups": [],
        "accessEntryArn": "arn:aws: eks:region-code:111122223333:access-
entry/my-cluster/role/111122223333/my-console-viewer-role/
abc12345-1234-1234-1234-123456789012",
        "createdAt": "2024-03-15T10:30:45.123000-07:00",
        "modifiedAt": "2024-03-15T10:30:45.123000-07:00",
        "tags": {},
        "username": "arn:aws: iam::111122223333:role/my-console-viewer-role",
        "type": "STANDARD"
    }
}
```

b. Associa una policy alla voce di accesso. Per visualizzare le risorse Kubernetes, utilizza: `AmazonEKSViewPolicy`

```
aws eks associate-access-policy \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy \
    --access-scope type=cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws: iam::111122223333:role/my-console-viewer-role",
    "associatedAt": "2024-03-15T10:31:15.456000-07:00"
}
```

Per un accesso specifico allo spazio dei nomi, puoi estendere la policy a namespace specifici:

```
aws eks associate-access-policy \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy \
    --access-scope type=namespace,namespaces=default,kube-system
```

c. Verifica che la voce di accesso sia stata creata correttamente:

```
aws eks describe-access-entry \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
```

d. Elenca le politiche associate per confermare l'associazione delle politiche:

```
aws eks list-associated-access-policies \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
```

Di seguito viene riportato un output di esempio.

```
{
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster"
            },
            "associatedAt": "2024-03-15T10:31:15.456000-07:00",
            "modifiedAt": "2024-03-15T10:31:15.456000-07:00"
        }
    ]
}
```

## CloudTrail visibilità

Durante la visualizzazione delle risorse Kubernetes, nei log verrà visualizzato il seguente nome dell'operazione: CloudTrail

- `AccessKubernetesApi`- Durante la lettura o la visualizzazione di risorse

Questo CloudTrail evento fornisce una traccia di controllo dell'accesso in lettura alle risorse Kubernetes.

> **Note**
>
> Questo nome dell'operazione viene visualizzato nei CloudTrail log solo a scopo di controllo. Non è un'azione IAM e non può essere utilizzata nelle dichiarazioni delle politiche IAM. Per controllare l'accesso in lettura alle risorse Kubernetes tramite le policy IAM, utilizza l'`eks:AccessKubernetesApi`autorizzazione come mostrato nella sezione. the section called "Autorizzazioni richieste"

# Concedi AWS ai servizi l'accesso in scrittura alle API Kubernetes

## Autorizzazioni richieste

Per consentire ai AWS servizi di eseguire operazioni di scrittura sulle risorse Kubernetes nel tuo cluster Amazon EKS, devi concedere sia le `eks:AccessKubernetesApi` autorizzazioni che quelle IAM. `eks:MutateViaKubernetesApi`

Ad esempio, Amazon SageMaker HyperPod utilizza queste autorizzazioni per abilitare la distribuzione di modelli da SageMaker AI Studio. Per ulteriori informazioni, consulta Configurare le autorizzazioni JavaScript SDK opzionali nella Amazon SageMaker AI Developer Guide.

> **Important**
>
> Le operazioni di scrittura come la creazione, l'aggiornamento e l'eliminazione richiedono entrambe le autorizzazioni: se una delle due autorizzazioni è mancante, le operazioni di scrittura avranno esito negativo.

## CloudTrail visibilità

Durante l'esecuzione di operazioni di scrittura sulle risorse Kubernetes, nei log verranno visualizzati nomi di operazioni specifici: CloudTrail

- `createKubernetesObject`- Durante la creazione di nuove risorse
- `updateKubernetesObject`- Quando si modificano risorse esistenti
- `deleteKubernetesObject`- Quando si rimuovono risorse

Questi CloudTrail eventi forniscono audit trail dettagliati di tutte le modifiche apportate alle risorse Kubernetes.

> **Note**
>
> Questi nomi di operazioni vengono visualizzati nei CloudTrail log solo a scopo di controllo. Non sono azioni IAM e non possono essere utilizzate nelle dichiarazioni politiche IAM. Per controllare l'accesso in scrittura alle risorse Kubernetes tramite le policy IAM, utilizza l'`eks:MutateViaKubernetesApi`autorizzazione come mostrato nella sezione. the section called “Autorizzazioni richieste”

# Connettere kubectl a un cluster EKS creando un file kubeconfig

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS.

In questo argomento viene creato un file `kubeconfig` per il cluster (o ne viene aggiornato uno esistente).

Lo strumento della linea di comando `kubectl` utilizza le informazioni di configurazione nei file `kubeconfig` per comunicare con il server API di un cluster. Per ulteriori informazioni, consulta la sezione Organizing Cluster Access Using kubeconfig Files nella documentazione di Kubernetes.

Amazon EKS utilizza il comando `aws eks get-token` con `kubectl` per l'autenticazione del cluster. Per impostazione predefinita, la AWS CLI utilizza le stesse credenziali restituite con il comando seguente:

```
aws sts get-caller-identity
```

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called “Configura kubectl ed eksctl”:

- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come`yum`,`apt-get`, o Homebrew per macOS sono spesso diverse versioni inferiori all'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata AWS CloudShell potrebbe inoltre essere inferiore a quella più recente. Per aggiornarla, vedi Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Un ruolo o un utente IAM con l'autorizzazione a utilizzare l'azione API `eks:DescribeCluster` per il cluster che specifichi. Per ulteriori informazioni, consulta the section called “Identity-based politiche”. Se utilizzi un'identità del tuo provider OpenID Connectper accedere al cluster, consulta la sezione Using kubectl nella documentazione di Kubernetes per creare o aggiornare il file `kubeconfig`.

## Connettiti usando la console EKS () CloudShell

È possibile connettersi al cluster direttamente dall'utente Console di gestione AWS AWS CloudShell, senza alcuna configurazione locale.

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Scegli il nome del cluster a cui vuoi connetterti.
3. Nella pagina dei dettagli del cluster, scegli Connetti dalla barra delle azioni in alto a destra.
4. AWS CloudShell si apre con una `kubectl` configurazione preconfigurata per il cluster.

Una volta connesso, puoi eseguire `kubectl` i comandi immediatamente:

```
kubectl get nodes
kubectl get pods -A
```

CloudShell le sessioni includono `kubectl` la AWS CLI e le CloudShell utilità standard. Questa funzionalità supporta cluster con endpoint di server API pubblici e privati. Per i cluster con endpoint privati, avvia CloudShell automaticamente un ambiente VPC per raggiungere il server API privato del cluster e richiede di inserire un nome per l'ambiente.

# `Crea il` file kubeconfig usando AWS CLI

- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come`yum`,`apt-get`, o Homebrew per macOS sono spesso diverse versioni inferiori all'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata AWS CloudShell potrebbe inoltre essere inferiore a quella più recente. Per aggiornarla, vedi Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Autorizzazione a utilizzare l'operazione API `eks:DescribeCluster` per il cluster specificato. Per ulteriori informazioni, consulta the section called "Identity-based politiche".

1. Crea o aggiorna un file `kubeconfig` per il cluster. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscilo *`my-cluster`* con il nome del cluster.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Per impostazione predefinita, il file di configurazione risultante viene creato nel percorso `kubeconfig` predefinito (`.kube`) nella home directory o unito a un file `config` esistente in quel percorso. È possibile specificare un altro percorso con l'opzione `--kubeconfig`.

   Puoi specificare un ARN del ruolo IAM con l'opzione `--role-arn` da utilizzare per l'autenticazione quando esegui comandi `kubectl`. Altrimenti, viene utilizzata l'entità IAM nella catena di credenziali AWS CLI o SDK predefinita. Puoi visualizzare la tua identità AWS CLI o SDK predefinita eseguendo il comando. `aws sts get-caller-identity`

   Per tutte le opzioni disponibili, esegui il `aws eks update-kubeconfig help` comando o consulta https://docs.aws.amazon.com/cli/latest/reference/eks/update-kubeconfig.html update-kubeconfig nella CLI Command Reference. AWS

2. Prova la configurazione.

   ```
   kubectl get svc
   ```

   Di seguito viene riportato un output di esempio:

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   ```

```
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called “Accesso non autorizzato o negato (kubectl) ” nell'argomento relativo alla risoluzione dei problemi.

# Concedi ai carichi di lavoro Kubernetes l'accesso a AWS utilizzando gli account di servizio Kubernetes

Un account di servizio Kubernetes fornisce un'identità per i processi eseguiti in un Pod. Per ulteriori informazioni, consulta Gestire gli account di servizio nella documentazione di Kubernetes. Se il tuo Pod deve accedere ai AWS servizi, puoi mappare l'account del servizio a un' AWS identità Identity and Access Management per concedere tale accesso. Per ulteriori informazioni, consulta the section called “Credenziali con IRSA” o the section called “Pod Identity”.

## Token dell'account di servizio

La BoundServiceAccountTokenVolumefunzionalità è abilitata per impostazione predefinita nelle versioni di Kubernetes. Questa funzionalità migliora la sicurezza dei token dell'account di servizio, consentendo ai carichi di lavoro in esecuzione su Kubernetes di richiedere token Web JSON associati a pubblico, tempo e chiavi. I token dell'account di servizio scadono entro un'ora, Nelle versioni precedenti di Kubernetes, i token non avevano scadenza. Di conseguenza, i client che si basano su questi token devono aggiornarli entro un'ora. Gli SDK client Kubernetes seguenti aggiornano automaticamente i token entro il periodo di tempo richiesto:

- Go versione `0.15.7` e successive
- Python versione `12.0.0` e successive
- Java versione `9.0.0` e successive
- JavaScript versione e successive `0.10.3`
- Ramo `master` di Ruby
- Haskell versione `0.3.0.0`
- Versione C# `7.0.5` e successive

Se il carico di lavoro utilizza una versione precedente del client, è necessario aggiornarla. Per consentire una migrazione agevole dei client verso i più recenti token dell'account di servizio con

limite di tempo, Kubernetes proroga il periodo di scadenza dei token, rispetto a quello predefinito di un'ora. Per i cluster Amazon EKS, il periodo di scadenza è prorogato a 90 giorni. Il server API Kubernetes del cluster Amazon EKS rifiuta le richieste con token che sono più vecchi di 90 giorni. Consigliamo di controllare le applicazioni e le relative dipendenze per assicurarti che gli SDK del client Kubernetes corrispondano o siano successivi alle versioni elencate sopra.

Quando il server API riceve richieste con token più vecchie di un'ora, le annota nel log eventi di controllo API con `annotations.authentication.k8s.io/stale-token`. Il valore dell'annotazione è simile a quello riportato di seguito:

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

Se nel cluster è abilitata la registrazione del piano di controllo (control-plane), le annotazioni di trovano nei registri di controllo. Puoi utilizzare la seguente query CloudWatch Logs Insights per identificare tutti i Pod nel tuo cluster Amazon EKS che utilizzano token obsoleti:

```
fields @timestamp
|filter @logStream like /kube-apiserver-audit/
|filter @message like /seconds after warning threshold/
|parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

`subject` si riferisce all'account di servizio utilizzato dal pod. `elapsedtime` indica in secondi il tempo trascorso dalla lettura dell'ultimo token. Le richieste al server API vengono negate quando `elapsedtime` supera i 90 giorni (7.776.000 secondi). Per utilizzare una delle versioni elencate sopra e aggiornare automaticamente il token, devi aggiornare in modo proattivo l'SDK client Kubernetes delle applicazioni. Se il token dell'account di servizio utilizzato è prossimo ai 90 giorni e non si ha tempo sufficiente per aggiornare le versioni dell'SDK client prima della scadenza, è possibile terminare i pod esistenti e crearne di nuovi. Ciò comporta una ridefinizione del token dell'account di servizio, concedendo agli utenti altri 90 giorni per aggiornare gli SDK della versione client.

Se il pod fa parte di un'implementazione, è consigliabile terminare i pod eseguendo un rollout in modo da mantenere un'elevata disponibilità. Per eseguire questa operazione, utilizzare il comando seguente. Sostituisci *`my-deployment`* con il nome della tua implementazione.

```
kubectl rollout restart deployment/my-deployment
```

## Componenti aggiuntivi del cluster

I componenti aggiuntivi del cluster seguenti sono stati aggiornati per utilizzare gli SDK client Kubernetes che riconfigurano automaticamente i token dell'account di servizio. Assicurati che le versioni elencate, o versioni successive, siano installate sul tuo cluster.

- Plugin helper per le metriche e plugin CNI di Amazon VPC per Kubernetes versione `1.8.0` e successive. Per verificare la versione attuale o aggiornarla, consulta the section called "CNI di Amazon VPC" e cni-metrics-helper.
- Versione CoredNS `1.8.4` e successive. Per verificare la versione attuale o aggiornarla, consulta the section called "CoreDNS".
- AWS Load Balancer Controller versione `2.0.0` e successive. Per verificare la versione attuale o aggiornarla, consulta the section called " AWS Controller Load Balancer".
- Versione `kube-proxy` attuale. Per verificare la versione attuale o aggiornarla, consulta the section called "kube-proxy".
- AWS per la versione Fluent Bit `2.25.0` o successiva. Per aggiornare la versione corrente, consulta Releases on GitHub.
- Versione dell'immagine di Fluentd 1.14.6-1.2 o versione successiva e plugin di filtro Fluentd per la versione dei metadati Kubernetes 2.11.1 o successive.

## Concessione AWS Autorizzazioni di Identity and Access Management per i carichi di lavoro sui cluster Amazon Elastic Kubernetes Service

Amazon EKS offre due modi per concedere le autorizzazioni di AWS Identity and Access Management ai carichi di lavoro eseguiti nei cluster Amazon EKS: ruoli IAM per gli account di servizio e EKS Pod Identities.

Ruoli IAM per gli account di servizio

> IAM roles for service accounts (IRSA) configura le applicazioni Kubernetes in esecuzione AWS con autorizzazioni IAM granulari per accedere a varie altre risorse come bucket Amazon AWS S3, tabelle Amazon DynamoDB e altro ancora. Puoi eseguire più applicazioni contemporaneamente nello stesso cluster Amazon EKS e assicurarti che ogni applicazione disponga solo del set minimo di autorizzazioni di cui ha bisogno. IRSA è stato creato per supportare diverse opzioni di implementazione di Kubernetes supportate da Amazon AWS EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on e cluster Kubernetes autogestiti su AWS istanze Amazon EC2.

Pertanto, IRSA è stato creato utilizzando AWS servizi fondamentali come IAM e non ha assunto alcuna dipendenza diretta dal servizio Amazon EKS e dall'API EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

Identità EKS Pod

EKS Pod Identity offre agli amministratori dei cluster un flusso di lavoro semplificato per l'autenticazione delle applicazioni per accedere a varie altre AWS risorse come bucket Amazon S3, tabelle Amazon DynamoDB e altro ancora. EKS Pod Identity è solo per EKS e, di conseguenza, semplifica il modo in cui gli amministratori dei cluster possono configurare le applicazioni Kubernetes per ottenere le autorizzazioni IAM. Queste autorizzazioni possono ora essere facilmente configurate con meno passaggi direttamente tramite l' Console di gestione AWS API EKS e la AWS CLI e non è necessario intraprendere alcuna azione all'interno del cluster in nessun oggetto Kubernetes. Gli amministratori del cluster non devono passare dai servizi EKS ai servizi IAM o utilizzare operazioni IAM privilegiate per configurare le autorizzazioni richieste dalle applicazioni. I ruoli IAM possono ora essere utilizzati su più cluster senza la necessità di aggiornare la policy di attendibilità dei ruoli durante la creazione di nuovi cluster. Le credenziali IAM fornite da EKS Pod Identity includono tag di sessione dei ruoli, con attributi come nome del cluster, namespace, nome dell'account di servizio. I tag di sessione dei ruoli consentono agli amministratori di creare un singolo ruolo in grado di funzionare su più account di servizio, consentendo l'accesso alle risorse in base ai AWS tag corrispondenti. Per ulteriori informazioni, consulta the section called "Pod Identity".

## Confronto tra EKS Pod Identity e IRSA

Ad alto livello, sia EKS Pod Identity che IRSA consentono di concedere autorizzazioni IAM alle applicazioni in esecuzione su cluster Kubernetes. Ma sono fondamentalmente diversi nel modo in cui vengono configurati, nei limiti supportati e nelle funzionalità abilitate. Di seguito, confrontiamo alcuni degli aspetti chiave di entrambe le soluzioni.

> **Note**
>
> AWS consiglia di utilizzare EKS Pod Identities per concedere l'accesso alle risorse ai pod ogni volta che AWS è possibile. Per ulteriori informazioni, consulta the section called "Pod Identity".

| Attributo | EKS Pod Identity | IRSA |
| --- | --- | --- |
| Estensibilità dei ruoli | È necessario configurare ogni ruolo una volta per instaurare un rapporto di attendibilità con il principale di servizio Amazon EKS introdotto di recente, `pods.eks.amazonaws.com`. Dopo questo passaggio una tantum, non è necessario aggiornare la policy di attendibilità del ruolo ogni volta che è utilizzato in un nuovo cluster. | È necessario aggiornare la policy di attendibilità del ruolo IAM con il nuovo endpoint del provider OIDC del cluster EKS ogni volta che si desidera utilizzare il ruolo in un nuovo cluster. |
| Scalabilità del cluster | EKS Pod Identity non richiede agli utenti di configurare il provider OIDC di IAM, quindi questo limite non si applica. | Ciascun cluster EKS ha un URL emittente OpenID Connect (OIDC) associato ad esso. Per utilizzare IRSA, è necessario creare un provider OpenID Connect univoco per ogni cluster EKS in IAM. IAM ha un limite globale predefinito di 100 provider OIDC per ogni account. AWS Se prevedi di avere più di 100 cluster EKS per ogni AWS account con IRSA, raggiungerai il limite del provider IAM OIDC. |
| Scalabilità dei ruoli | EKS Pod Identity non richiede agli utenti di definire una relazione di attendibilità tra il ruolo IAM e l'account di servizio nella policy di attendibilità, quindi questo limite non è applicato. | In IRSA, definisci la relazione di attendibilità tra un ruolo IAM e un account di servizio nella policy di attendibilità del ruolo. Per impostazione predefinita, la lunghezza della dimensione della policy |

| Attributo | EKS Pod Identity | IRSA |
| --- | --- | --- |
| | | di attendibilità è 2048. Ciò significa che in genere è possibile definire 4 relazioni di attendibilità in un'unica policy di attendibilità. Sebbene sia possibile aumentare il limite di lunghezza della policy di attendibilità, in genere si è limitati a un massimo di 8 relazioni di attendibilità all'interno di una singola policy di attendibilità. |
| Utilizzo della quota API di STS | EKS Pod Identity semplifica l'invio delle AWS credenziali ai pod e non richiede il codice per effettuare chiamate direttamente con il AWS Security Token Service (STS). Il servizio EKS gestisce l'assunzione dei ruoli e fornisce le credenziali alle applicazioni scritte utilizzando l' AWS SDK nei pod senza che i pod comunichino con STS o utilizzino STS API Quota. AWS | In IRSA, le applicazioni scritte utilizzando l' AWS SDK nei pod utilizzano i token per chiamare l'API sul Security Token Service (STS). `AssumeRoleWithWebIdentity` AWS A seconda della logica del codice sull' AWS SDK, è possibile che il codice effettui chiamate non necessarie a AWS STS e riceva errori di limitazione. |

| Attributo | EKS Pod Identity | IRSA |
| --- | --- | --- |
| Riutilizzabilità dei ruoli | AWS Le credenziali temporanee STS fornite da EKS Pod Identity includono tag di sessione del ruolo, come il nome del cluster, lo spazio dei nomi, il nome dell'account del servizio. I tag di sessione di ruolo consenton o agli amministratori di creare un singolo ruolo IAM che può essere utilizzato con più account di servizio, con diverse autorizzazioni effettive , consentendo l'accesso alle AWS risorse in base ai tag ad essi allegati. Questo ruolo è chiamato anche controllo degli accessi basato su attributi (ABAC). Per ulteriori informazi oni, consulta the section called "Concessione dell'accesso dei pod". | AWS I tag di sessione STS non sono supportati. È possibile riutilizzare un ruolo tra i cluster, ma ogni pod riceve tutte le autorizzazioni del ruolo. |
| Ambienti supportati | EKS Pod Identity è disponibile solo su Amazon EKS. | È possibile utilizzare IRSA come Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on AWS e cluster Kubernetes autogestiti su istanze Amazon EC2. |

| Attributo | EKS Pod Identity | IRSA |
| --- | --- | --- |
| Versioni EKS supportate | Tutte le versioni del cluster EKS supportate. Per le versioni delle piattaforme specifiche, consulta the section called "Versioni del cluster EKS Pod Identity". | Tutte le versioni del cluster EKS supportate. |

## Ruoli IAM per gli account di servizio

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Le applicazioni nei contenitori di un Pod possono utilizzare un AWS SDK o la AWS CLI per effettuare richieste API ai AWS servizi utilizzando le autorizzazioni IAM ( AWS Identity and Access Management). Le applicazioni devono firmare le proprie richieste AWS API con credenziali. AWS I ruoli IAM per gli account di servizio (IRSA) forniscono la possibilità di gestire le credenziali per le applicazioni, analogamente al modo in cui i profili di istanza Amazon EC2 forniscono le credenziali alle istanze Amazon EC2. Invece di creare e distribuire le AWS credenziali ai contenitori o utilizzare il ruolo dell'istanza Amazon EC2, associ un ruolo IAM a un account di servizio Kubernetes e configura i tuoi Pods per utilizzare l'account di servizio. Non puoi utilizzare i ruoli IAM per gli account di servizio con cluster locali per Amazon EKS su Outposts. AWS

La caratteristica dei ruoli IAM per gli account di servizio offre i seguenti vantaggi:

- Privilegio minimo: è possibile definire l'ambito delle autorizzazioni IAM per un account di servizio; solo i pod che utilizzano tale account avranno accesso alle autorizzazioni definite. Questa caratteristica elimina anche la necessità di soluzioni di terze parti, ad esempio `kiam` o `kube2iam`.
- Isolamento delle credenziali: quando l'accesso al servizio di metadati di istanza (IMDS)di Amazon EC2 è limitato, un container del pod può recuperare solo le credenziali per il ruolo IAM associato all'account del servizio che utilizza. Un container non ha mai accesso alle credenziali utilizzate da altri container in altri pod. Se IMDS non è soggetto a restrizioni, i container del pod hanno accesso anche al ruolo IAM del nodo Amazon EKS potrebbero essere in grado di accedere

alle credenziali dei ruoli IAM di altri pod sullo stesso nodo. Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

> **Note**
>
> I pod configurati con `hostNetwork: true` avranno sempre accesso IMDS, ma gli AWS SDK e la CLI utilizzeranno le credenziali IRSA quando abilitati.

- Verificabilità: l'accesso e la registrazione degli eventi sono disponibili tramite per garantire un controllo retrospettivo. AWS CloudTrail

> **Important**
>
> I container non costituiscono un limite di sicurezza e l'uso dei ruoli IAM per gli account di servizio non cambia questa situazione. I pod assegnati allo stesso nodo condivideranno un kernel e potenzialmente altre risorse a seconda della configurazione del pod. Sebbene i pod in esecuzione su nodi separati siano isolati a livello di elaborazione, esistono applicazioni di nodo che dispongono di autorizzazioni aggiuntive nell'API Kubernetes oltre all'ambito di una singola istanza. Alcuni esempi sono `kubelet`, `kube-proxy`, i driver di archiviazione CSI o le tue applicazioni Kubernetes.

Abilita i ruoli IAM per gli account di servizio completando le seguenti procedure:

1. Creazione di un provider OIDC IAM per il cluster: questa operazione viene completata solo una sola volta per cluster.

   > **Note**
   >
   > Se il VPC del cluster non ha accesso a Internet in uscita e non hai impostato l'accesso privato all'endpoint OIDC del cluster, le operazioni che raggiungono quell'endpoint dall'interno del VPC, come la creazione di un provider OIDC con, non possono risolvere il nome host dell'emittente OIDC. `eksctl` Segue un messaggio di errore di esempio:
   >
   > ```
   > server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
   > ```

Per raggiungere l'endpoint OIDC del cluster privatamente dal tuo VPC, crea un endpoint di interfaccia VPC per esso (`com.amazonaws.region-code.oidc-eks`) con DNS privato abilitato. Per ulteriori informazioni, consulta the section called " AWS PrivateLink".
In alternativa, puoi eseguire il comando all'esterno del VPC (ad esempio, in) o creare un resolver condizionale a orizzonte diviso. AWS CloudShell Per un esempio, consulta la richiesta di funzionalità di Amazon EKS su. https://github.com/aws/containers-roadmap/issues/2038 GitHub

2. Assegna i ruoli IAM agli account del servizio Kubernetes: completa questa procedura per ogni set univoco di autorizzazioni che desideri abbia un'applicazione.
3. Configura i Pod per utilizzare un account di servizio Kubernetes: completa questa procedura per ogni Pod che deve accedere ai servizi. AWS
4. Usa IRSA con l' AWS SDK: conferma che il carico di lavoro utilizzi un AWS SDK di una versione supportata e che il carico di lavoro utilizzi la catena di credenziali predefinita.

## Informazioni di base su IAM, Kubernetes e OpenID Connect (OIDC)

Nel 2014, AWS Identity and Access Management ha aggiunto il supporto per le identità federate utilizzando OpenID Connect (OIDC). Questa funzionalità consente di autenticare le chiamate AWS API con i provider di identità supportati e ricevere un token web JSON (JWT) OIDC valido. Puoi passare questo token all'operazione `AssumeRoleWithWebIdentity` API AWS STS e ricevere le credenziali di ruolo temporanee IAM. Puoi utilizzare queste credenziali per interagire con qualsiasi AWS servizio, inclusi Amazon S3 e DynamoDB.

Ogni token JWT è firmato da una coppia di chiavi di firma. Le chiavi vengono fornite dal provider OIDC gestito da Amazon EKS e la chiave privata ruota ogni 7 giorni. Amazon EKS conserva le chiavi pubbliche fino alla loro scadenza. Se si connettono client OIDC esterni, si tenga presente che è necessario aggiornare le chiavi di firma prima della scadenza della chiave pubblica. Scopri Fetch signing keys to validate OIDC tokens.

Kubernetes utilizza da tempo gli account di servizio come sistema di identità interno. I pod possono eseguire l'autenticazione con il server API Kubernetes utilizzando un token con montaggio automatico (che è un token JWT non OIDC) che solo il server API Kubernetes può convalidare. Questi token legacy dell'account di servizio non hanno scadenza e la rotazione della chiave di firma è un processo difficile. Nella versione `1.12` di Kubernetes, è stato aggiunto il supporto per una nuova

funzionalità `ProjectedServiceAccountToken`. Questa funzionalità è un token Web OIDC JSON che contiene anche l'identità dell'account del servizio e supporta un pubblico configurabile.

Amazon EKS ora ospita un endpoint di rilevamento OIDC pubblico per ogni cluster, contenente le chiavi di firma per i token Web `ProjectedServiceAccountToken` JSON, in modo che i sistemi esterni, ad esempio IAM, possano convalidare e accettare i token OIDC emessi da Kubernetes.

## Per creare un provider di identità IAM OIDC per il cluster

Il cluster ha un URL emittente OpenID Connect (OIDC) associato. Per utilizzare i ruoli IAM ( AWS Identity and Access Management) per gli account di servizio, deve esistere un provider IAM OIDC per l'URL dell'emittente OIDC del cluster.

Prerequisiti

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo oppure. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come`yum`,`apt-get`, o Homebrew per macOS sono spesso diverse versioni inferiori all'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata AWS CloudShell potrebbe inoltre essere inferiore a quella più recente. Per aggiornarla, vedi Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

È possibile creare un gestore OIDC IAM per il cluster utilizzando `eksctl` o la Console di gestione AWS.

Creare un gestore OIDC (eksctl)

1. La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Determinare l'ID dell'emittente OIDC per il cluster.

   Recuperare gli ID dell'emittente OIDC del cluster e archiviarlo in una variabile. Sostituire `<my-cluster>` con il proprio valore.

   ```
   cluster_name=<my-cluster>
   oidc_id=$(aws eks describe-cluster --name $cluster_name --query
    "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
   echo $oidc_id
   ```
3. Determinare se un gestore OIDC IAM con l'ID del cluster è già presente nell'account.

   ```
   aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
   ```

   Se viene restituito un output, il cluster dispone già di un gestore OIDC IAM ed è possibile saltare al passaggio successivo. Se non viene restituito alcun output, devi creare un provider OIDC IAM per il cluster.
4. Creare un provider di identità OIDC IAM per il cluster con il comando seguente.

   ```
   eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
   ```

   > **Note**
   >
   > Se il VPC del cluster non ha accesso a Internet in uscita e non hai impostato l'accesso privato all'endpoint OIDC del cluster, le operazioni che raggiungono quell'endpoint dall'interno del VPC, come la creazione di un provider OIDC con, non possono risolvere il nome host dell'emittente OIDC. `eksctl` Segue un messaggio di errore di esempio:
   >
   > ```
   > ** server cant find oidc.eks.<region-code>.amazonaws.com: NXDOMAIN
   > ```

Per raggiungere l'endpoint OIDC del cluster privatamente dal tuo VPC, crea un endpoint di interfaccia VPC per esso (`com.amazonaws.<region-code>.oidc-eks`) con DNS privato abilitato. Per ulteriori informazioni, consulta Accedere all'endpoint OIDC del cluster utilizzando. AWS PrivateLink

In alternativa, puoi eseguire il comando all'esterno del VPC (ad esempio, in AWS CloudShell) o creare un resolver condizionale split-horizon. Per un esempio, consulta la richiesta di funzionalità di Amazon EKS su. https://github.com/aws/containers-roadmap/issues/2038 GitHub

Crea un provider OIDC (AWS Console)

1. Aprire la Console Amazon EKS.
2. Nel riquadro a sinistra, seleziona Cluster, quindi seleziona il nome del cluster nella pagina Cluster.
3. Nella sezione Dettagli della scheda Panoramica, annota il valore dell'URL del provider OpenID Connect.
4. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
5. Nel pannello di navigazione, scegli Identity Providers (Provider di identità) in Access management (Gestione accesso). Se un Provider nell'elenco corrisponde all'URL del cluster, significa che si dispone già di un provider per il cluster. Se nell'elenco non è presente un gestore che corrisponde all'URL del cluster, è necessario crearne uno.
6. Per creare un provider, selezionare Aggiungi provider.
7. Per Tipo di provider, selezionare OpenID Connect.
8. Per URL provider, inserire l'URL del gestore OIDC del cluster.
9. Per Destinatari, inserisci `sts.amazonaws.com`
10. (Facoltativo) Aggiungere eventuali tag, ad esempio un tag per identificare quale cluster è assegnato a questo gestore.
11. Scegli Aggiungi provider.

Fase successiva: the section called "Assegnazione di un ruolo IAM"

## Assegnare ruoli IAM agli account di servizio Kubernetes

Questo argomento spiega come configurare un account di servizio Kubernetes per assumere un ruolo AWS Identity and Access Management (IAM). Tutti i Pod configurati per utilizzare l'account di

servizio possono quindi accedere a qualsiasi AWS servizio a cui il ruolo dispone delle autorizzazioni per accedere.

Prerequisiti

- Un cluster esistente. Se non ne possiedi già uno, crealo seguendo una delle guide in Nozioni di base.
- Creare un provider di identità OpenID Connect (OIDC) IAM per il cluster. Per sapere se disponi di un account o, per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

Fase 1: creare la policy IAM

Se desideri associare una policy IAM esistente al ruolo IAM, passa al prossimo passaggio.

1. Creare una policy IAM Puoi creare la tua policy o copiare una policy AWS gestita che già concede alcune delle autorizzazioni di cui hai bisogno e personalizzarla in base alle tue esigenze specifiche. Per ulteriori informazioni, consulta Creating IAM policies nella Guida per l'utente di IAM.
2. Crea un file che includa le autorizzazioni per i AWS servizi a cui desideri che i tuoi Pod accedano. Per un elenco di tutte le azioni per tutti i AWS servizi, consulta il Service Authorization Reference.

Puoi eseguire il comando seguente per creare un file policy di esempio che consenta l'accesso in sola lettura a un bucket Amazon S3. Facoltativamente, puoi archiviare le informazioni di configurazione o uno script di bootstrap in questo bucket; i container nel pod possono leggere il file dal bucket e caricarlo nell'applicazione. Se desideri creare questa policy di esempio, copia i seguenti contenuti sul dispositivo. Sostituiscilo *my-pod-secrets-bucket* con il nome del tuo bucket ed esegui il comando.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

3. Creare la policy IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

Fase 2: creare e associare il ruolo IAM

Creare un ruolo IAM e associarlo a un account di servizio Kubernetes. È possibile utilizzare uno dei due `eksctl` o la AWS CLI.

Creare e associare il ruolo (eksctl)

Questo comando `eksctl` crea un account di servizio Kubernetes nel namespace specificato, crea un ruolo IAM (se non esiste) con il nome specificato, allega un ARN della policy IAM esistente al ruolo e annota l'account di servizio con l'ARN del ruolo IAM. Assicurati di sostituire i valori segnaposto di esempio in questo comando con i valori specifici. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster
 my-cluster --role-name my-role \
    --attach-policy-arn arn:aws: iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> Se l'account del ruolo o del servizio esiste già, il comando precedente potrebbe non riuscire. `eksctl`dispone di diverse opzioni che è possibile fornire in tali situazioni. Per ulteriori informazioni, esegui `eksctl create iamserviceaccount --help`.

Crea e associa il ruolo (AWS CLI)

Se disponi di un account di servizio Kubernetes esistente che deve assumere un ruolo IAM, puoi saltare questa fase.

1. Creare un account di servizio Kubernetes. Copia i seguenti contenuti sul dispositivo. Sostituisci *`my-service-account`* con il nome desiderato e *`default`* con un namespace diverso, se necessario. Se si modifica*`default`*, lo spazio dei nomi deve già esistere.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

2. Imposta l'ID AWS del tuo account su una variabile di ambiente con il seguente comando.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

3. Imposta il provider di identità OIDC del cluster su una variabile di ambiente con il comando seguente. Sostituisci *`my-cluster`* con il nome del cluster.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
```

4. Imposta le variabili per lo spazio dei nomi e il nome dell'account del servizio. Sostituisci *`my-service-account`* con l'account del servizio Kubernetes che desideri assumere il ruolo. Sostituisci *`default`* con lo spazio dei nomi dell'account del servizio.

```
export namespace=default
```

```
export service_account=my-service-account
```

5. Per creare una policy di attendibilità del ruolo IAM, esegui il comando seguente. Se desideri consentire a tutti gli account di servizio all'interno di un namespace di utilizzare il ruolo, copia i seguenti contenuti sul dispositivo. Sostituisci *`StringEquals`* con `StringLike` e sostituisci *`$service_account`* con. * Puoi aggiungere più voci nelle condizioni `StringEquals` e `StringLike` riportate di seguito per utilizzare più account di servizio o namespace con il ruolo. Per consentire ai ruoli di un AWS account diverso da quello in cui si trova il cluster di assumere il ruolo, consulta the section called "IAM multi-account" per ulteriori informazioni.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
```

6. Crea il ruolo. Sostituisci *`my-role`* con un nome per il ruolo IAM e *`my-role-description`* con una descrizione per il tuo ruolo.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

7. Allegare una policy IAM al ruolo. Sostituisci *`my-role`* con il nome del ruolo IAM e *`my-policy`* con il nome di una policy esistente che hai creato.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws: iam::
$account_id:policy/my-policy
```

8. Annota l'account del servizio con il nome della risorsa Amazon (ARN) del ruolo IAM che desideri che l'account del servizio assuma. Sostituisci *my-role* con il nome del ruolo IAM esistente. Supponiamo di aver consentito a un ruolo di un AWS account diverso da quello in cui si trova il cluster di assumere il ruolo in un passaggio precedente. Quindi, assicurati di specificare l' AWS account e il ruolo dell'altro account. Per ulteriori informazioni, consulta the section called "IAM multi-account".

```
kubectl annotate serviceaccount -n $namespace $service_account eks.amazonaws.com/
role-arn=arn:aws: iam::$account_id:role/my-role
```

9. (Facoltativo) Configura l'endpoint del servizio AWS Security Token per un account di servizio. AWS consiglia di utilizzare un endpoint AWS STS regionale anziché un endpoint globale. Ciò riduce la latenza, fornisce una ridondanza integrata e aumenta la validità del token di sessione.

Fase 3: confermare la configurazione

1. Confermare che la policy di attendibilità del ruolo IAM sia configurata correttamente.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

   Di seguito viene riportato un output di esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-service-
account",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
```

```
            }
        }
    ]
}
```

2. Conferma che la policy che hai associato al tuo ruolo in un passaggio precedente sia associata al ruolo.

```
aws iam list-attached-role-policies --role-name my-role --query
 "AttachedPolicies[].PolicyArn" --output text
```

Di seguito viene riportato un output di esempio.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Imposta una variabile per memorizzare il nome della risorsa Amazon (ARN) della policy che desideri utilizzare. Sostituiscilo *my-policy* con il nome della politica per cui desideri confermare le autorizzazioni.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Visualizza la versione predefinita della policy.

```
aws iam get-policy --policy-arn $policy_arn
```

Di seguito viene riportato un output di esempio.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Visualizza il contenuto della policy per assicurarti che includa tutte le autorizzazioni di cui il pod ha bisogno. Se necessario, sostituiscilo *1* nel comando seguente con la versione restituita nell'output precedente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Di seguito viene riportato un output di esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Se hai creato la policy di esempio in un passaggio precedente, il risultato è lo stesso. Se hai creato una politica diversa, il *example* contenuto è diverso.

6. Confermare che l'account di servizio Kubernetes sia annotato con il ruolo.

```
kubectl describe serviceaccount my-service-account -n default
```

Di seguito viene riportato un output di esempio.

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-
role
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

Fasi successive

- the section called "Assegnazione a pod"

## Configurazione di pod per utilizzare un account di servizio Kubernetes

Se un Pod deve accedere ai AWS servizi, devi configurarlo per utilizzare un account di servizio Kubernetes. L'account di servizio deve essere associato a un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni per accedere ai servizi AWS .

- Un cluster esistente. Se non se ne possiede già uno, crearlo utilizzando una delle guide in Nozioni di base.
- Creare un provider di identità OpenID Connect (OIDC) IAM per il cluster. Per sapere se disponi di un account o, per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Un account di servizio Kubernetes esistente associato a un ruolo IAM. L'account del servizio deve essere annotato con il nome della risorsa Amazon (ARN) del ruolo IAM. Al ruolo deve essere associata una policy IAM che contenga le autorizzazioni che desideri che i tuoi Pod abbiano per utilizzare i servizi. AWS Per ulteriori informazioni su come creare e configurare l'account e il ruolo del servizio, consulta the section called "Assegnazione di un ruolo IAM".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

  1. Utilizza il seguente comando per creare un manifesto di implementazione per implementare un pod con cui confermare la configurazione. Sostituire i valori di esempio con i propri valori.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Implementa il file manifesto al cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Verifica che le variabili di ambiente richieste esistano per i pod.

   a. Visualizza i pod distribuiti con l'implementazione nella fase precedente.

   ```
   kubectl get pods | grep my-app
   ```

   Di seguito viene riportato un output di esempio.

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Visualizza l'ARN del ruolo IAM che il pod sta utilizzando.

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
   ```

   Di seguito viene riportato un output di esempio.

   ```
   AWS_ROLE_ARN: arn:aws: iam::111122223333:role/my-role
   ```

Il ruolo ARN deve corrispondere al ruolo ARN con cui hai annotato l'account del servizio esistente. Per ulteriori informazioni sull'annotazione dell'account del servizio, consulta the section called "Assegnazione di un ruolo IAM".

c. Conferma che il pod abbia un file di token di identità web montato.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
```

Di seguito viene riportato un output di esempio.

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/serviceaccount/
token
```

`kubelet` richiede e memorizza il token per conto del pod. Per impostazione predefinita, la `kubelet` aggiorna il token se è più vecchio dell'80% del suo tempo totale di vita o se il token è più vecchio di 24 ore. È possibile modificare la durata di scadenza per qualsiasi account, ad eccezione dell'account di servizio predefinito, con le impostazioni nelle specifiche del pod. Per ulteriori informazioni, consultare Proiezione Volume Token Acount di Servizio nella documentazione di Kubernetes.

Il Webhook di Pod Identity di Amazon EKS sugli orologi del cluster per pod che utilizzano un account di servizio con la seguente annotazione:

```
eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-role
```

Il webhook applica le variabili di ambiente precedenti a questi pod. Il cluster non deve utilizzare il webhook per configurare le variabili di ambiente e i montaggi del file di token. È possibile configurare manualmente i pod per ottenere queste variabili di ambiente. Le versioni supportate dell' AWS SDK cercano innanzitutto queste variabili di ambiente nel provider della catena di credenziali. Le credenziali del ruolo sono utilizzate per i pod che soddisfano questi criteri.

4. Verifica che i tuoi Pod possano interagire con i AWS servizi utilizzando le autorizzazioni assegnate nella policy IAM associata al tuo ruolo.

> **Note**
>
> Quando un Pod utilizza AWS le credenziali di un ruolo IAM associato a un account di servizio, la AWS CLI o SDKs altro nei contenitori per quel Pod utilizza le credenziali fornite da quel ruolo. Se non si limita l'accesso alle credenziali fornite al ruolo IAM del nodo di Amazon EKS, il pod ha comunque accesso a tali credenziali. Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se il tuo pod non riesce a interagire con i servizi come previsto, completa i seguenti passaggi per confermare che tutto sia configurato correttamente.

a. Verifica che i tuoi Pod utilizzino una versione AWS SDK che supporti l'assunzione di un ruolo IAM tramite un file di token di identità web OpenID Connect. Per ulteriori informazioni, consulta the section called "SDK supportati".

b. Conferma che l'implementazione stia utilizzando l'account del servizio.

```
kubectl describe deployment my-app | grep "Service Account"
```

Di seguito viene riportato un output di esempio.

```
Service Account:  my-service-account
```

c. Se i tuoi pod non riescono ancora ad accedere ai servizi, controlla i passaggi descritti in Assign IAM roles to Kubernetes service accounts per confermare che il ruolo e l'account del servizio siano configurati correttamente.

## Configura il AWS Endpoint Security Token Service per un account di servizio

Se utilizzi un account di servizio Kubernetes con ruoli IAM per gli account di servizio, puoi configurare il tipo di endpoint AWS Security Token Service utilizzato dall'account di servizio.

AWS consiglia di utilizzare gli endpoint AWS STS regionali anziché l'endpoint globale. Ciò riduce la latenza, fornisce una ridondanza integrata e aumenta la validità del token di sessione. Il AWS Security Token Service deve essere attivo nella AWS regione in cui è in esecuzione il Pod. Inoltre, l'applicazione deve avere una ridondanza integrata per una AWS regione diversa in caso di guasto

del servizio nella AWS regione. Per ulteriori informazioni, consulta Managing AWS STS in an AWS Region nella IAM User Guide.

- Un cluster esistente. Se non se ne possiede già uno, crearlo utilizzando una delle guide in Nozioni di base.
- Provider OIDC IAM esistente per il cluster. Per ulteriori informazioni, consulta the section called "Gestore di identità OIDC IAM".
- Un account del servizio Kubernetes esistente configurato per l'uso con la funzionalità Ruoli IAM di Amazon EKS per gli account di servizio.

> **Note**
>
> Per rendere private le richieste di credenziali IRSA in un VPC senza uscita Internet, crea un endpoint di interfaccia AWS STS (`com.amazonaws.region-code.sts`) e imposta l'endpoint regionale come descritto in questo argomento. L'endpoint globale () bypassa l'endpoint VPC. `sts.amazonaws.com` Per raggiungere l' discovery/JWKS endpoint OIDC del cluster anche privatamente (per la configurazione del provider e i validatori di token), crea un endpoint VPC separato (). `com.amazonaws.region-code.oidc-eks` Per ulteriori informazioni, consulta the section called "Accedi all'endpoint OIDC del cluster utilizzando AWS PrivateLink".

Tutti gli esempi seguenti utilizzano l'account di servizio Kubernetes aws-node utilizzato dal plugin CNI di Amazon VPC. Puoi sostituirlo *`example values`* con i tuoi account di servizio, pod, namespace e altre risorse.

1. Selezionare un pod che utilizza un account di servizio per il quale si desidera modificare l'endpoint. Determina in quale AWS regione viene eseguito il Pod. Sostituiscilo *`aws-node-6mfgv`* con il nome del tuo Pod e *`kube-system`* con il namespace del tuo Pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   Di seguito viene riportato un output di esempio.

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   Nell'output precedente, il Pod è in esecuzione su un nodo nella regione AWS us-west-2.

2. Determinare il tipo di endpoint utilizzato dall'account di servizio del pod.

```
kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

Di seguito viene riportato un output di esempio.

```
AWS_STS_REGIONAL_ENDPOINTS: regional
```

Se l'endpoint attuale è globale, l'output restituisce `global`. Se non viene restituito alcun output, il tipo di endpoint predefinito è in uso e non è stato sovrascritto.

3. Puoi cambiare il tipo di endpoint utilizzato dal tuo account di servizio dal tipo predefinito a un tipo diverso con uno dei seguenti comandi. Sostituisci *`aws-node`* con il nome dell'account del servizio e *`kube-system`* con lo spazio dei nomi dell'account del servizio.

- Se il tipo di endpoint predefinito o attuale è globale e vuoi modificarlo in regionale:

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
```

Se si utilizzano i ruoli IAM per gli account di servizio per generare URL S3 prefirmati nell'applicazione in esecuzione nei container dei pod, il formato dell'URL per gli endpoint Regionali è simile al seguente esempio:

```
https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
```

- Se il tipo di endpoint predefinito o attuale è regionale e vuoi modificarlo in globale:

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
```

Se l'applicazione effettua richieste esplicite agli endpoint globali AWS STS e non sovrascrivi il comportamento predefinito di utilizzo degli endpoint regionali nei cluster Amazon EKS, le richieste avranno esito negativo e verrà generato un errore. Per ulteriori informazioni, consulta the section called "I container dei pod riceveranno il seguente errore: `An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`".

Se si utilizzano i ruoli IAM per gli account di servizio per generare URL S3 prefirmati nell'applicazione in esecuzione nei container dei pod, il formato dell'URL per gli endpoint globali è simile al seguente esempio:

```
https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/
us-west-2/s3/aws4_request&...
```

Se disponi di un'automazione che prevede l'URL prefirmato in un determinato formato o se la tua applicazione o le dipendenze downstream che utilizzano URL prefirmati hanno aspettative per la AWS regione target, apporta le modifiche necessarie per utilizzare l'endpoint STS appropriato. AWS

4. Eliminare e ricreare i pod esistenti associati all'account di servizio per applicare le variabili di ambiente delle credenziali. Il web hook di modifica non le applica ai pod già in esecuzione. Puoi sostituire *Pods* e *-l k8s-app=aws-node* con le informazioni relative ai Pod per i quali hai impostato l'annotazione. *kube-system*

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Conferma che tutti i Pod sono stati riavviati.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Visualizzare le variabili di ambiente per uno dei pod. Verifica che il valore `AWS_STS_REGIONAL_ENDPOINTS` sia quello che hai impostato in una fase precedente.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

Di seguito viene riportato un output di esempio:

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

## Eseguire l'autenticazione a un altro account con IRSA

# possibile configurare le autorizzazioni IAM multi-account creando un gestore di identità dal cluster di un altro account o utilizzando operazioni `AssumeRole` concatenate. Negli esempi seguenti,

l'account A possiede un cluster Amazon EKS che supporta i ruoli IAM per gli account di servizio. I pod in esecuzione sul cluster devono assumere le autorizzazioni IAM dall'account B.

- L'opzione 1 è più semplice ma richiede l'Account B per creare e gestire un provider di identità OIDC per il cluster dell'Account A.
- L'opzione 2 mantiene la gestione OIDC nell'Account A ma richiede il concatenamento dei ruoli tramite due chiamate. `AssumeRole`

Opzione 1: creare un provider di identità dal cluster di un altro account

In questo esempio, l'account A fornisce all'account B l'URL dell'emittente OpenID Connect (OIDC) dal relativo cluster. L'account B segue le istruzioni indicate alle pagine Create an IAM OIDC provider for your cluster e the section called "Assegnazione di un ruolo IAM" utilizzando l'URL dell'emittente OIDC dal cluster dell'account A. Quindi, un amministratore del cluster annota l'account di servizio nel cluster dell'Account A per utilizzare il ruolo dell'Account B (*444455556666*).

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::444455556666:role/account-b-role
```

Opzione 2: utilizzare operazioni **`AssumeRole`** concatenate

In questo approccio, ogni account crea un ruolo IAM. Il ruolo dell'Account B si affida all'Account A e il ruolo dell'Account A utilizza la federazione OIDC per ottenere le credenziali dal cluster. Il Pod concatena quindi i due ruoli utilizzando i AWS profili CLI.

Fase 1: Creare il ruolo di destinazione nell'Account B

L'account B (*444455556666*) crea un ruolo IAM con le autorizzazioni necessarie ai pod del cluster dell'account A. L'account B associa la politica di autorizzazione desiderata a questo ruolo, quindi aggiunge la seguente politica di fiducia.

Politica di fiducia per il ruolo dell'Account B: questa politica consente al ruolo IRSA specifico dell'Account A di assumere questo ruolo.

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

**Important**

Per ottenere il minimo privilegio, sostituisci l'`Principal`ARN con il ruolo specifico ARN dell'account A invece di usare l'account root (). `arn:aws:iam::111122223333:root` L'utilizzo dell'account root consente a qualsiasi principale IAM dell'account A di assumere questo ruolo.

Fase 2: Creare il ruolo IRSA nell'Account A

L'account A (*111122223333*) crea un ruolo con una politica di fiducia che ottiene le credenziali dal provider di identità creato con l'indirizzo dell'emittente OIDC del cluster.

Politica di fiducia per il ruolo dell'Account A (federazione OIDC): questa politica consente al provider OIDC del cluster EKS di emettere credenziali per questo ruolo.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

```
        }
      }
    }
  ]
}
```

> **Important**
>
> Per ottenere il privilegio minimo, aggiungi una `StringEquals` condizione per la `sub` dichiarazione di limitare questo ruolo a uno specifico account di servizio Kubernetes. Senza una `sub` condizione, qualsiasi account di servizio nel cluster può assumere questo ruolo. Il `sub` valore utilizza il formato`system:serviceaccount:`*NAMESPACE*`:`*SERVICE_ACCOUNT_NAME* . Ad esempio, per limitarsi a un account di servizio denominato `my-service-account` nel `default` namespace:
>
> ```
> "oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
>  "system:serviceaccount:default:my-service-account"
> ```

Passaggio 3: allegare l' AssumeRole autorizzazione al ruolo dell'Account A

L'Account A allega una politica di autorizzazione al ruolo creato nella Fase 2. Questa politica consente al ruolo di assumere il ruolo dell'Account B.

Politica di autorizzazione per il ruolo dell'Account A: questa politica concede `sts:AssumeRole` all'Account B il ruolo obiettivo.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

Passaggio 4: configura il Pod per concatenare i ruoli

Il codice dell'applicazione per i pod che assumono il ruolo dell'account B utilizza due profili: `account_b_role` e `account_a_role`. Il profilo `account_b_role` utilizza il profilo `account_a_role` come origine. Per la AWS CLI, il `~/.aws/config` file è simile al seguente.

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws: iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws: iam::111122223333:role/account-a-role
```

Per specificare profili concatenati per altri AWS SDKs, consulta la documentazione dell'SDK che stai utilizzando. Per ulteriori informazioni, consulta Strumenti su cui costruire. AWS

## Usa IRSA con AWS SDK

Utilizzo delle credenziali

Per utilizzare le credenziali di IAM roles for service accounts (IRSA), il codice può utilizzare qualsiasi AWS SDK per creare un client per un AWS servizio con un SDK e, per impostazione predefinita, l'SDK cerca in una catena di posizioni le credenziali di AWS Identity and Access Management da utilizzare. I ruoli IAM per le credenziali degli account di servizio verranno utilizzati se non specifichi un provider di credenziali quando crei il client o inizializzi in altro modo l'SDK.

Questo perché i ruoli IAM per gli account di servizio sono stati aggiunti come passaggio nella catena di credenziali predefinita. Se i tuoi carichi di lavoro attualmente utilizzano credenziali che si trovano in una fase precedente della catena di credenziali, tali credenziali continueranno a essere utilizzate anche se configuri i ruoli IAM per gli account di servizio per lo stesso carico di lavoro.

L'SDK scambia automaticamente il token OIDC dell'account di servizio con credenziali temporanee da Security Token Service utilizzando l'azione. AWS `AssumeRoleWithWebIdentity` Amazon EKS e questa azione SDK continuano a ruotare le credenziali temporanee rinnovandole prima della scadenza.

Quando si utilizzano i ruoli IAM per gli account di servizio, i contenitori dei Pod devono utilizzare una versione AWS SDK che supporti l'assunzione di un ruolo IAM tramite un file token di identità Web OpenID Connect. Assicurati di utilizzare le seguenti versioni, o successive, per il tuo SDK: AWS

- Java (versione 2) — 2.10.11
- Java — 1.12.782
- AWS SDK per Go v1 — 1.23.13
- AWS SDK for Go v2: supporto per tutte le versioni
- Python (Boto3) — 1.9.220
- Python (botocore) — 1.12.200
- AWS CLI — 1.16.232
- Nodo: 2.525.0 e 3.27.0
- Ruby — 3.58.0
- C++ — 1.7.174
- .NET: 3,3.659.1 – Devi includere anche `AWSSDK.SecurityToken`.
- PHP — 3.110.7

Molti popolari componenti aggiuntivi di Kubernetes, come Cluster Autoscaler, Route internet traffic with Load AWS Balancer Controller e il plug-in Amazon VPC CNI per Kubernetes, supportano i ruoli IAM per gli account di servizio.

Per assicurarti che l'SDK utilizzato sia supportato, quando crei i container segui la procedura di installazione per il tuo SDK preferito disponibile in Strumenti per costruire su AWS.

Considerazioni

Java

Quando si utilizza Java, è necessario includere il modulo `sts` nel percorso della classe. Per ulteriori WebIdentityTokenFileCredentialsProviderinformazioni, consulta la documentazione di Java SDK.

## Recupera le chiavi di firma per convalidare i token OIDC

Kubernetes rilascia un `ProjectedServiceAccountToken` a ciascun account di servizio Kubernetes. Questo token è un token OIDC, che è un tipo di token web JSON (JWT). Amazon EKS ora ospita un endpoint pubblico OIDC per ogni cluster che contiene le chiavi di firma per i token in modo che i sistemi esterni possano convalidarlo.

Per convalidare un `ProjectedServiceAccountToken`, devi recuperare le chiavi di firma pubbliche OIDC, chiamate anche set di chiavi Web JSON (JWKS). Utilizza queste chiavi nell'applicazione per

convalidare il token. Ad esempio, puoi usare PyJWT Python library per convalidare i token utilizzando queste chiavi. Per ulteriori informazioni su `ProjectedServiceAccountToken`, consultare the section called "Informazioni di base su IAM, Kubernetes e OpenID Connect (OIDC)".

> **Note**
>
> Se il validatore viene eseguito in un VPC senza accesso a Internet in uscita, puoi recuperare il documento di scoperta e JWKS privatamente creando un endpoint di interfaccia per l'endpoint OIDC del cluster (). AWS PrivateLink `com.amazonaws.region-code.oidc-eks` Per ulteriori informazioni, consulta the section called "Accedi all'endpoint OIDC del cluster utilizzando AWS PrivateLink".

Prerequisiti

- Un provider OpenID Connect (OIDC) di AWS Identity and Access Management (IAM) esistente per il tuo cluster. Per determinare se si dispone già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- AWS CLI: uno strumento a riga di comando per lavorare con i AWS servizi, incluso Amazon EKS. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando. Dopo aver installato la AWS CLI, ti consigliamo di configurarla anche tu. Per ulteriori informazioni, consulta Configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.

Procedura

1. Recupera l'URL OIDC per il tuo cluster Amazon EKS utilizzando l'interfaccia a riga di comando. AWS

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. Recupera la chiave di firma pubblica usando curl o uno strumento simile. Il risultato è un JSON Web Key Set (JWKS).

> **Important**
>
> Amazon EKS limita le chiamate all'endpoint OIDC. È necessario caricare nella cache la chiave di firma pubblica. Rispetta l'intestazione `cache-control` inclusa nella risposta.

> **Important**
>
> Amazon EKS ruota la chiave di firma OIDC ogni sette giorni.

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}]
```

## Scopri come EKS Pod Identity concede ai pod l'accesso a AWS services

Le applicazioni nei contenitori di un Pod possono utilizzare un AWS SDK o la AWS CLI per effettuare richieste API AWS ai servizi AWS utilizzando le autorizzazioni Identity and Access Management (IAM). Le applicazioni devono firmare le proprie richieste AWS API con credenziali. AWS

Le associazioni EKS Pod Identity forniscono la possibilità di gestire le credenziali per le applicazioni, analogamente al modo in cui i profili di istanza Amazon EC2 forniscono le credenziali alle istanze Amazon EC2. Invece di creare e distribuire AWS le tue credenziali ai contenitori o utilizzare il ruolo dell'istanza Amazon EC2, associ un ruolo IAM a un account di servizio Kubernetes e configuri i tuoi Pods per utilizzare l'account di servizio.

Ogni associazione EKS Pod Identity associa un ruolo a un account di servizio in un namespace nel cluster specificato. Se hai la stessa applicazione in più cluster, puoi creare associazioni identiche in ogni cluster senza modificare la policy di attendibilità del ruolo.

Se un pod utilizza un account di servizio con un'associazione, Amazon EKS imposta le variabili di ambiente nei container del pod. Le variabili di ambiente configurano gli AWS SDK, inclusa la AWS CLI, per utilizzare le credenziali EKS Pod Identity.

## Vantaggi delle associazioni EKS Pod Identity

Le associazioni EKS Pod Identity offrono i seguenti vantaggi:

- Privilegio minimo: è possibile definire l'ambito delle autorizzazioni IAM per un account di servizio; solo i pod che utilizzano tale account avranno accesso alle autorizzazioni definite. Questa caratteristica elimina anche la necessità di soluzioni di terze parti, ad esempio `kiam` o `kube2iam`.
- Isolamento delle credenziali: quando l'accesso al servizio di metadati di istanza (IMDS)di Amazon EC2 è limitato, un container del pod può recuperare solo le credenziali per il ruolo IAM associato all'account del servizio che utilizza. Un container non ha mai accesso alle credenziali utilizzate da altri container in altri pod. Se IMDS non è soggetto a restrizioni, i container del pod hanno accesso anche al ruolo IAM del nodo Amazon EKS potrebbero essere in grado di accedere

alle credenziali dei ruoli IAM di altri pod sullo stesso nodo. Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

> **Note**
>
> I pod configurati con `hostNetwork: true` avranno sempre accesso IMDS, ma gli AWS SDK e la CLI utilizzeranno le credenziali Pod Identity quando abilitati.

- Verificabilità: la registrazione degli accessi e degli eventi è disponibile per facilitare il controllo retrospettivo. AWS CloudTrail

> **Important**
>
> I container non sono un limite di sicurezza e l'uso di Pod Identity non modifica questo aspetto. I pod assegnati allo stesso nodo condivideranno un kernel e potenzialmente altre risorse a seconda della configurazione del pod. Sebbene i pod in esecuzione su nodi separati siano isolati a livello di elaborazione, esistono applicazioni di nodo che dispongono di autorizzazioni aggiuntive nell'API Kubernetes oltre all'ambito di una singola istanza. Alcuni esempi sono `kubelet`, `kube-proxy`, driver di archiviazione CSI o le tue applicazioni Kubernetes.

EKS Pod Identity è un metodo più semplice di the section called "Credenziali con IRSA", in quanto non utilizza provider di identità OIDC. EKS Pod Identity presenta i seguenti miglioramenti:

- Operazioni indipendenti: in molte organizzazioni, la creazione di provider di identità OIDC è una responsabilità di team diversi rispetto all'amministrazione dei cluster Kubernetes. EKS Pod Identity offre una netta separazione dei compiti, in cui tutta la configurazione delle associazioni EKS Pod Identity viene eseguita in Amazon EKS e tutta la configurazione delle autorizzazioni IAM viene eseguita in IAM.
- Riusabilità: EKS Pod Identity utilizza un singolo principale IAM anziché principali separati per ogni cluster utilizzato dai ruoli IAM per gli account di servizio. L'amministratore IAM aggiunge il seguente principale alla policy di attendibilità di qualsiasi ruolo per renderlo utilizzabile da parte delle associazioni EKS Pod Identity.

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
```

```
        }
```

- Scalabilità: ogni set di credenziali temporanee viene assunto dal servizio EKS Auth in EKS Pod Identity, anziché da ogni AWS SDK eseguito in ciascun pod. Quindi, si esegue Amazon EKS Pod Identity Agent su ogni nodo rilasciando le credenziali agli SDK. Pertanto, il carico si riduce a una volta per ciascun nodo e non è duplicato in ciascun pod. Per ulteriori dettagli del processo, consulta the section called "Come funziona".

Per ulteriori informazioni su come confrontare le due alternative, consulta the section called "Accesso al carico di lavoro a AWS ".

## Panoramica sulla configurazione delle associazioni EKS Pod Identity

Attiva le associazioni EKS Pod Identity completando le seguenti procedure:

1. the section called "Configurazione dell'agente": completi questa procedura una sola volta per ogni cluster. Non è necessario completare questo passaggio se la modalità automatica EKS è abilitata sul cluster.
2. the section called "Assegnazione di un ruolo IAM": completa questa procedura per ciascun set univoco di autorizzazioni che desideri abbia un'applicazione.
3. the section called "Account di servizio dei pod"— Completate questa procedura per ogni Pod che necessita dell'accesso ai servizi. AWS
4. the section called "SDK supportati"— Conferma che il carico di lavoro utilizzi un AWS SDK di una versione supportata e che utilizzi la catena di credenziali predefinita.

## Limits

- Sono supportate fino a 5.000 associazioni EKS Pod Identity per cluster per mappare i ruoli IAM agli account di servizio Kubernetes.

## Considerazioni

- IAM Role Association: ogni account di servizio Kubernetes in un cluster può essere associato a un ruolo IAM dello stesso account del cluster. AWS Per modificare il ruolo, modifica l'associazione EKS Pod Identity. Per l'accesso multi-account, delega l'accesso al ruolo utilizzando i ruoli IAM. Per saperne di più, consulta Delegare l'accesso tra AWS account utilizzando i ruoli IAM nella Guida per l'utente IAM.

- Agente EKS Pod Identity: l'agente Pod Identity è necessario per utilizzare EKS Pod Identity. L'agente funziona come Kubernetes `DaemonSet` sui nodi del cluster, fornendo credenziali solo ai pod sullo stesso nodo. Utilizza le porte occupanti del `hostNetwork` nodo `80` e `2703` l'indirizzo locale del collegamento (`169.254.170.23`per IPv4, per IPv6). `[fd00:ec2::23]` Se IPv6 è disabilitato nel cluster, disabilita IPv6 per l'agente Pod Identity. Per ulteriori informazioni, consulta Disabilitare IPv6 nell'agente EKS Pod Identity.
- Coerenza finale: le associazioni EKS Pod Identity alla fine sono coerenti, con potenziali ritardi di diversi secondi dopo le chiamate API. Evita di creare o aggiornare associazioni in percorsi di codice critici e ad alta disponibilità. Esegui, invece, queste azioni in routine di inizializzazione o configurazione separate e meno frequenti. Per ulteriori informazioni, consulta Gruppi di sicurezza per pod nella Guida alle best practice EKS.
- Considerazioni sui proxy e sui gruppi di sicurezza: per i pod che utilizzano un proxy, aggiungi `169.254.170.23` (IPv4) e `[fd00:ec2::23]` (IPv6) alle variabili di ambiente `no_proxy/NO_PROXY` per evitare richieste non riuscite all'agente EKS Pod Identity. Se utilizzi Security Groups for Pods con il CNI AWS VPC, imposta `ENABLE_POD_ENI` il flag su «true» e `POD_SECURITY_GROUP_ENFORCING_MODE` il flag su «standard». Per ulteriori informazioni, consulta Assegnare di gruppi di sicurezza ai singoli pod.

### Versioni del cluster EKS Pod Identity

Per utilizzare le associazioni EKS Pod Identity, il cluster deve avere una versione della piattaforma uguale o successiva a quella elencata nella tabella seguente o una versione Kubernetes successiva a quelle elencate nella tabella. Per trovare la versione consigliata di Amazon EKS Pod Identity Agent per una versione Kubernetes, consulta the section called "Verifica la compatibilità".

| Versione di Kubernetes | Versione della piattaforma |
|---|---|
| Versioni di Kubernetes non elencate | Tutte le versioni delle piattaforma supportate |
| `1.28` | `eks.4` |

### Restrizioni di EKS Pod Identity

Le associazioni EKS Pod Identity sono disponibili in:

- Versioni del cluster Amazon EKS elencate nell'argomento precedente the section called "Versioni del cluster EKS Pod Identity".

- Nodi worker nel cluster che sono istanze Amazon EC2 di Linux.

Le associazioni EKS Pod Identity non sono disponibili in quanto segue:

- AWS Outposts.
- Amazon EKS Anywhere.
- Cluster Kubernetes creati ed eseguiti su Amazon EC2. I componenti di EKS Pod Identity sono disponibili solo su Amazon EKS.

Non è possibile utilizzare le associazioni EKS Pod Identity con:

- Pod che vengono eseguiti ovunque, tranne che su istanze Amazon EC2 di Linux. I pod Linux e Windows che funzionano su AWS Fargate (Fargate) non sono supportati. I pod che vengono eseguiti su istanze Amazon EC2 di Windows non sono supportati.

## Comprensione di come funziona EKS Pod Identity

Le associazioni Amazon EKS Pod Identity offrono la possibilità di gestire le credenziali per le applicazioni, in modo simile a come i profili di istanza di Amazon EC2 forniscono le credenziali alle istanze Amazon EC2.

Amazon EKS Pod Identity fornisce le credenziali per i tuoi carichi di lavoro con un'API EKS Auth aggiuntiva e un pod di agenti che viene eseguito su ogni nodo.

Nei tuoi componenti aggiuntivi, come i componenti aggiuntivi Amazon EKS e i controller, gli operatori e altri componenti aggiuntivi autogestiti, l'autore deve aggiornare il proprio software per utilizzare gli SDK più recenti. AWS Per l'elenco della compatibilità tra EKS Pod Identity e i componenti aggiuntivi prodotti da Amazon EKS, consulta la sezione precedente the section called “Restrizioni di EKS Pod Identity”.

Utilizzo delle associazioni EKS Pod Identity nel codice

Nel tuo codice, puoi utilizzare gli SDK per accedere ai servizi. AWS AWS Scrivi codice per creare un client per un AWS servizio con un SDK e, per impostazione predefinita, l'SDK cerca in una catena di posizioni le credenziali di AWS Identity and Access Management da utilizzare. Dopo aver trovato credenziali valide, la ricerca viene interrotta. Per ulteriori informazioni sulle posizioni predefinite utilizzate, consulta la catena di fornitori di credenziali nella Guida di riferimento agli AWS SDK and Tools.

Le associazioni EKS Pod Identity sono state aggiunte al provider di credenziali del container, che viene cercato in un passaggio nella catena di credenziali predefinita. Se i carichi di lavoro utilizzano attualmente credenziali che si trovano all'inizio della catena di credenziali, quest'ultime continuano a essere utilizzate anche se configuri un'associazione EKS Pod Identity per lo stesso carico di lavoro. In questo modo è possibile migrare in sicurezza da altri tipi di credenziali creando l'associazione prima di rimuovere le vecchie credenziali.

Il provider di credenziali del container assegna credenziali temporanee da un agente che viene eseguito su ogni nodo. In Amazon EKS, l'agente è Amazon EKS Pod Identity Agent e su Amazon Elastic Container Service l'agente è `amazon-ecs-agent`. Gli SDK utilizzano variabili di ambiente per individuare l'agente a cui connettersi.

Al contrario, i ruoli IAM per gli account di servizio forniscono un token di identità Web che l' AWS SDK deve scambiare con AWS Security Token Service utilizzando. `AssumeRoleWithWebIdentity`

Come funziona un agente EKS Pod Identity con un pod

1. Quando Amazon EKS avvia un nuovo pod che utilizza un account di servizio con un'associazione EKS Pod Identity, il cluster aggiunge il seguente contenuto al manifesto:

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. Kubernetes seleziona su quale nodo eseguire il pod. Quindi, Amazon EKS Pod Identity Agent sul nodo utilizza l'AssumeRoleForPodIdentityazione per recuperare credenziali temporanee dall'API EKS Auth.

3. EKS Pod Identity Agent rende disponibili queste credenziali per gli AWS SDK che esegui all'interno dei contenitori.
4. Utilizza l'SDK nell'applicazione senza specificare un provider di credenziali per utilizzare la catena di credenziali predefinita. In alternativa, specifica il provider di credenziali del container. Per ulteriori informazioni sulle posizioni predefinite utilizzate, consulta la catena di fornitori di credenziali nella Guida di riferimento agli AWS SDK e agli strumenti.
5. L'SDK utilizza le variabili di ambiente per connettersi all'EKS Pod Identity Agent e recuperare le credenziali.

> **Note**
>
> Se i carichi di lavoro attualmente utilizzano credenziali che si trovano all'inizio della catena di credenziali, tali credenziali continueranno a essere utilizzate anche se configuri un'associazione EKS Pod Identity per lo stesso carico di lavoro.

## Configurazione dell'agente Amazon EKS Pod Identity

Le associazioni Amazon EKS Pod Identity offrono la possibilità di gestire le credenziali per le applicazioni, in modo simile a come i profili di istanza di Amazon EC2 forniscono le credenziali alle istanze Amazon EC2.

Amazon EKS Pod Identity fornisce le credenziali per i tuoi carichi di lavoro con un'API EKS Auth aggiuntiva e un pod di agenti che viene eseguito su ogni nodo.

> **Tip**
>
> Non è necessario installare l'agente EKS Pod Identity sui cluster in modalità automatica EKS. Questa funzionalità è integrata in modalità automatica EKS.

### Considerazioni

- Per impostazione predefinita, l'agente EKS Pod Identity è preinstallato sui cluster in modalità automatica EKS. Per ulteriori informazioni, consulta Modalità automatica EKS.
- Per impostazione predefinita, l'agente EKS Pod Identity ascolta su un indirizzo `IPv4` e sull'indirizzo `IPv6` perché i pod richiedano le credenziali. L'agente utilizza l'indirizzo IP di loopback (localhost) `169.254.170.23` per `IPv4` e l'indirizzo IP localhost `[fd00:ec2::23]` per `IPv6`.

- Se disabiliti gli indirizzi `IPv6` o impedisci in altro modo gli indirizzi IP `IPv6` localhost, l'agente non può avviarsi. Per avviare l'agente su nodi che non possono utilizzare `IPv6`, segui i passaggi indicati in the section called "Disabilitazione di IPv6" per disabilitare la configurazione `IPv6`.

## Creazione di Amazon EKS Pod Identity Agent

### Prerequisiti dell'agente

> **Important**
>
> I nodi su cui viene eseguito EKS Pod Identity Agent devono poter accedere all'API di autenticazione EKS. Se si utilizzano sottoreti private per i nodi, è necessario configurare un endpoint di AWS PrivateLink interfaccia per l'API EKS Auth. Per ulteriori informazioni, consulta the section called " AWS PrivateLink".

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base. La versione del cluster e la versione della piattaforma devono essere uguali o successive alle versioni elencate nelle versioni del cluster EKS Pod Identity.
- Il ruolo del nodo dispone delle autorizzazioni per consentire all'agente di eseguire l'azione `AssumeRoleForPodIdentity` nell'API EKS Auth. Puoi utilizzare la politica AWS gestita: Amazon EKSWorker NodePolicy o aggiungere una politica personalizzata simile alla seguente:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

Questa azione può essere limitata dai tag per ridurre i ruoli che possono essere assunti dai pod che utilizzano l'agente.

- I nodi possono raggiungere e scaricare immagini da Amazon ECR. L'immagine del container per il componente aggiuntivo si trova nei registri elencati in Visualizza registri delle immagini del container Amazon per i componenti aggiuntivi Amazon EKS.

  Tieni presente che puoi modificare la posizione dell'immagine e fornire `imagePullSecrets` componenti aggiuntivi EKS nelle impostazioni di configurazione opzionali nella e nella `--configuration-values` AWS CLI. Console di gestione AWS

Agente di configurazione con console AWS

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi scegli il nome del cluster per cui configurare il componente aggiuntivo di EKS Pod Identity Agent.
3. Selezionare la scheda Componenti aggiuntivi.
4. Scegli Ottieni altri componenti aggiuntivi.
5. Seleziona la casella nella parte superiore destra di quella del componente aggiuntivo relativo a EKS Pod Identity Agent e scegli Avanti.
6. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona una versione qualsiasi nell'elenco a discesa Versione.
7. (Facoltativo) Espandi Impostazioni di configurazione facoltative per inserire una configurazione aggiuntiva. Ad esempio, puoi fornire una posizione alternativa per l'immagine del container e `ImagePullSecrets`. Lo Schema JSON con le chiavi accettate sono mostrate in Schema di configurazione del componente aggiuntivo.

   Inserisci le chiavi e i valori di configurazione in Valori di configurazione.
8. Scegli Next (Successivo).
9. Verifica che i pod di EKS Pod Identity Agent siano in esecuzione sul cluster.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Di seguito viene riportato un output di esempio.

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1     Running
   1 (24h ago)   24h
```

Ora puoi utilizzare le associazioni EKS Pod Identity nel cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di un ruolo IAM".

Agente di configurazione con AWS CLI

1. Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --
addon-version v1.0.0-eksbuild.1
```

> **Note**
>
> L'agente EKS Pod Identity non utilizza `service-account-role-arn` per ruoli IAM per account di servizio. È necessario fornire a EKS Pod Identity Agent le autorizzazioni nel ruolo del nodo.

2. Verifica che i pod di EKS Pod Identity Agent siano in esecuzione sul cluster.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Di seguito viene riportato un output di esempio.

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1     Running
   1 (24h ago)   24h
```

Ora puoi utilizzare le associazioni EKS Pod Identity nel cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di un ruolo IAM".

## Assegnazione di un ruolo IAM a un account di servizio Kubernetes

Questo argomento spiega come configurare un account di servizio Kubernetes per assumere un ruolo di AWS Identity and Access Management (IAM) con EKS Pod Identity. Tutti i Pod configurati per utilizzare l'account di servizio possono quindi accedere a qualsiasi AWS servizio a cui il ruolo dispone delle autorizzazioni di accesso.

Per creare un'associazione EKS Pod Identity, è sufficiente un solo passaggio; si crea l'associazione in EKS tramite AWS CLI, AWS SDK AWS CloudFormation e altri strumenti. Console di gestione AWS Non sono presenti dati o metadati sulle associazioni all'interno del cluster in nessun oggetto Kubernetes e non si aggiungono annotazioni agli account di servizio.

Prerequisiti

- Un cluster esistente. Se non se ne possiede già uno, crearlo seguendo una delle guide in Nozioni di base.
- Il principale IAM che sta creando l'associazione deve avere `iam:PassRole`.
- La versione più recente della AWS CLI installata e configurata sul dispositivo o. AWS CloudShell È possibile verificare la versione corrente con `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

Crea un'associazione Pod Identity (AWS Console)

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui configurare il componente aggiuntivo di EKS Pod Identity Agent.
3. Scegli la scheda Accesso.
4. In Associazioni Pod Identity, scegli Crea.
5. Per Ruolo IAM, seleziona il ruolo IAM con le autorizzazioni che desideri assegnare al carico di lavoro.

> **Note**
>
> L'elenco contiene solo i ruoli con la seguente policy di attendibilità che consente a EKS Pod Identity di utilizzarli.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`: EKS Pod Identity utilizza `AssumeRole` per assumere il ruolo IAM prima di passare le credenziali temporanee ai pod.

`sts:TagSession`— EKS Pod Identity utilizza `TagSession` per includere i tag di sessione nelle richieste a AWS STS.

È possibile utilizzare questi tag nelle chiavi di condizione nella policy di attendibilità per limitare gli account di servizio, i namespace e i cluster che possono utilizzare questo ruolo.

Per un elenco di chiavi di condizione di Amazon EKS, consultare Condizioni per Amazon Elastic Kubernetes Service in Service Authorization Reference. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

6. Per Namespace Kubernetes, seleziona il namespace Kubernetes che contiene l'account di servizio e il carico di lavoro. Facoltativamente, è possibile specificare un namespace che non esiste nel cluster.

7. Per l'account del servizio Kubernetes, seleziona l'account del servizio Kubernetes da utilizzare. Il manifesto del carico di lavoro Kubernetes deve specificare questo account di servizio. Facoltativamente, è possibile specificare un account di servizio in base al nome che non è presente nel cluster.
8. (Facoltativo) Seleziona Disabilita i tag di sessione per disabilitare i tag di sessione predefiniti che Pod Identity aggiunge automaticamente quando assume il ruolo.
9. (Facoltativo) Attiva Configure session policy per configurare una policy IAM per applicare restrizioni aggiuntive a questa associazione Pod Identity oltre alle autorizzazioni definite nella policy IAM allegata al ruolo IAM.

> **Note**
>
> Una policy di sessione può essere applicata solo quando l'impostazione Disabilita i tag di sessione è selezionata.

10. (Facoltativo) Per Tag, scegli Aggiungi tag per aggiungere metadati in una coppia chiave-valore. Questi tag vengono applicati all'associazione e possono essere utilizzati nelle policy IAM.

    Puoi ripetere questo passaggio per aggiungere più tag.
11. Scegli Create (Crea).

Crea un'associazione Pod Identity (AWS CLI)

1. Se desideri associare una policy IAM esistente al ruolo IAM, passa al prossimo passaggio.

   Creare una policy IAM È possibile creare una politica personalizzata o copiare una politica AWS gestita che già concede alcune delle autorizzazioni necessarie e personalizzarla in base ai requisiti specifici. Per ulteriori informazioni, consulta Creating IAM policies nella Guida per l'utente di IAM.

   a. Crea un file che includa le autorizzazioni per i AWS servizi a cui desideri che i tuoi Pod accedano. Per un elenco di tutte le azioni per tutti i AWS servizi, consulta il Service Authorization Reference.

      Puoi eseguire il comando seguente per creare un file policy di esempio che consenta l'accesso in sola lettura a un bucket Amazon S3. Facoltativamente, puoi archiviare le informazioni di configurazione o uno script di bootstrap in questo bucket; i container nel pod possono leggere il file dal bucket e caricarlo nell'applicazione. Se desideri creare questa policy di esempio, copia

i seguenti contenuti sul dispositivo. Sostituisci *my-pod-secrets-bucket* con il nome del tuo bucket ed esegui il comando.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

b. Creare la policy IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

2. Crea un ruolo IAM e associalo a un account del servizio Kubernetes.

a. Se disponi di un account di servizio Kubernetes esistente che desideri assuma un ruolo IAM, puoi saltare questo passaggio.

Creare un account di servizio Kubernetes. Copia i seguenti contenuti sul dispositivo. Sostituiscilo *my-service-account* con il nome desiderato e *default* con un namespace diverso, se necessario. Se si modifica*default*, lo spazio dei nomi deve già esistere.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

Eseguire il seguente comando seguente.

```
kubectl apply -f my-service-account.yaml
```

b. Per creare una policy di attendibilità del ruolo IAM, esegui il comando seguente.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

c. Crea il ruolo. Sostituisci *my-role* con un nome per il ruolo IAM e *my-role-description* con una descrizione per il tuo ruolo.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://
trust-relationship.json --description "my-role-description"
```

d. Allegare una policy IAM al ruolo. Sostituisci *my-role* con il nome del ruolo IAM e *my-policy* con il nome di una policy esistente che hai creato.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:
 iam::111122223333:policy/my-policy
```

> **Note**
>
> A differenza dei ruoli IAM per gli account di servizio, EKS Pod Identity non utilizza l'annotazione sull'account di servizio.

e. Esegui il comando seguente per creare l'associazione. Sostituiscilo `my-cluster` con il nome del cluster, sostituiscilo *my-service-account* con il nome desiderato e *default* con un namespace diverso, se necessario.

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-arn
 arn:aws: iam::111122223333:role/my-role --namespace default --service-account my-
service-account
```

Di seguito viene riportato un output di esempio.

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws: iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> È possibile specificare un namespace e un account di servizio in base al nome che non esiste nel cluster. È necessario creare il namespace, l'account di servizio e il carico di lavoro che utilizza l'account di servizio affinché l'associazione EKS Pod Identity funzioni.

Confermare la configurazione

1. Conferma che la policy di attendibilità del ruolo IAM sia configurata correttamente.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

Di seguito viene riportato un output di esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

2. Conferma che la policy che hai associato al tuo ruolo in un passaggio precedente sia associata al ruolo.

```
aws iam list-attached-role-policies --role-name my-role --query
 'AttachedPolicies[].PolicyArn' --output text
```

Di seguito viene riportato un output di esempio.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Imposta una variabile per memorizzare il nome della risorsa Amazon (ARN) della policy che desideri utilizzare. Sostituiscilo *my-policy* con il nome della policy per cui desideri confermare le autorizzazioni.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Visualizza la versione predefinita della policy.

```
aws iam get-policy --policy-arn $policy_arn
```

Di seguito viene riportato un output di esempio.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
```

```
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Visualizza il contenuto della policy per assicurarti che includa tutte le autorizzazioni di cui il pod ha bisogno. Se necessario, sostituiscilo *1* nel comando seguente con la versione restituita nell'output precedente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Di seguito viene riportato un output di esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Se hai creato la policy di esempio in un passaggio precedente, il risultato è lo stesso. Se hai creato una politica diversa, il *example* contenuto è diverso.

Fasi successive

the section called "Account di servizio dei pod"

## Accesso AWS Risorse che utilizzano i ruoli IAM di EKS Pod Identity Target

Quando esegui applicazioni su Amazon Elastic Kubernetes Service (Amazon EKS), potresti dover accedere AWS a risorse esistenti in account diversi. AWS Questa guida mostra come configurare l'accesso su più account utilizzando EKS Pod Identity, che consente ai pod Kubernetes di accedere ad altre risorse utilizzando i ruoli di destinazione. AWS

Prerequisiti

Prima di iniziare, assicurati di aver completato i passaggi seguenti:

- Configurazione dell'agente Amazon EKS Pod Identity
- Creazione di un ruolo di EKS Pod Identity

## Come funziona

Pod Identity consente alle applicazioni del cluster EKS di accedere alle AWS risorse tra gli account attraverso un processo chiamato concatenamento dei ruoli.

Quando crei un'associazione Pod Identity, puoi fornire due ruoli IAM: un ruolo EKS Pod Identity nello stesso account del cluster EKS e un ruolo IAM Target dall'account contenente AWS le risorse a cui desideri accedere (come i bucket S3 o i database RDS). Il ruolo EKS Pod Identity deve trovarsi nell'account del cluster EKS a causa PassRole dei requisiti IAM, mentre il ruolo IAM di Target può essere in qualsiasi account. AWS PassRole consente a un' AWS entità di delegare l'assunzione del ruolo a un altro servizio. EKS Pod Identity utilizza PassRole per connettere un ruolo a un account di servizio Kubernetes, richiedendo che sia il ruolo che l'identità che lo trasmette si trovino nello stesso AWS account del cluster EKS. Quando il pod dell'applicazione deve accedere alle AWS risorse, richiede le credenziali a Pod Identity. Quindi, Pod Identity esegue automaticamente due assunzione di ruolo in sequenza: prima assume il ruolo EKS Pod Identity, poi utilizza tali credenziali per assumere il ruolo IAM di destinazione. Questo processo fornisce al pod credenziali temporanee con le autorizzazioni definite nel ruolo di destinazione, consentendo l'accesso sicuro alle risorse di altri account. AWS

## Considerazioni sul caching

Per via dei meccanismi del caching, gli aggiornamenti per un ruolo IAM in un'associazione Pod Identity esistente potrebbero non avere effetto immediato nei pod in esecuzione sul cluster EKS. L'agente Pod Identity memorizza nella cache le credenziali IAM in base alla configurazione dell'associazione al momento del recupero delle credenziali. Se l'associazione include solo un ruolo EKS Pod Identity e nessun ruolo IAM di destinazione, le credenziali memorizzate nella cache durano 6 ore. Se l'associazione include sia l'ARN del ruolo EKS Pod Identity che un ruolo IAM di destinazione, le credenziali memorizzate nella cache durano 59 minuti. La modifica di un'associazione esistente, ad esempio l'aggiornamento dell'ARN del ruolo EKS Pod Identity, l'aggiunta di un ruolo Target IAM o l'aggiornamento della politica di sessione sull'associazione, non reimposta la cache esistente. Di conseguenza, l'agente non riconoscerà gli aggiornamenti fino all'aggiornamento delle credenziali memorizzate nella cache. Per applicare prima le modifiche, è possibile ricreare i pod esistenti; in caso contrario, sarà necessario attendere la scadenza della cache.

Passaggio 1: creazione e associazione di un ruolo IAM di destinazione

In questo passaggio, stabilirai una catena di affidabilità sicura creando e configurando un ruolo IAM di destinazione. A titolo dimostrativo, creeremo un nuovo ruolo Target IAM per stabilire una catena di fiducia tra due AWS account: il ruolo EKS Pod Identity (ad esempio`eks-pod-identity-primary-role`) nell' AWS account del cluster EKS ottiene l'autorizzazione ad assumere il ruolo Target IAM (ad esempio`eks-pod-identity-aws-resources`) nel tuo account di destinazione, abilitando l'accesso a AWS risorse come i bucket Amazon S3.

Creazione del ruolo IAM di destinazione

1. Apri la console IAM Amazon.
2. Nella barra di navigazione in alto, verifica di aver effettuato l'accesso all'account contenente AWS le risorse (come i bucket S3 o le tabelle DynamoDB) per il tuo ruolo IAM di Target.
3. Nel pannello di navigazione a sinistra, seleziona Ruoli.
4. Scegli il pulsante Crea ruolo, quindi l' AWS account nella sezione «Tipo di entità affidabile».
5. Scegli Altro AWS account, inserisci il tuo numero di AWS account (l'account in cui esiste il tuo ruolo EKS Pod Identity), quindi scegli Avanti.
6. Aggiungi le politiche di autorizzazione che desideri associare al ruolo (ad esempio AmazonS3FullAccess), quindi scegli Avanti.
7. Immetti un nome ruolo, come `MyCustomIAMTargetRole`, quindi scegli Crea ruolo.

Aggiorna la policy di affidabilità di un ruolo IAM di destinazione

1. Dopo aver creato il ruolo, ritornerai all'elenco Ruoli. Trova e seleziona il nuovo ruolo creato nel passaggio precedente (ad esempio, `MyCustomIAMTargetRole`).
2. Seleziona la scheda Relazioni di attendibilità.
3. Seleziona Modifica policy di attendibilità.
4. Nell'editor della policy, sostituisci il file JSON predefinito con la tua policy di affidabilità. Sostituisci i valori segnaposto per il nome del ruolo e `111122223333` nell'ARN del ruolo IAM con l'ID dell' AWS account che ospita il tuo cluster EKS. Opzionalmente, puoi anche utilizzare PrincipalTags la policy di fiducia dei ruoli per autorizzare solo account di servizio specifici di un determinato cluster e namespace ad assumere il tuo ruolo di destinazione. Esempio:

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/eks-cluster-arn": "arn:aws:eks:us-
east-1:111122223333:cluster/example-cluster",
          "aws:RequestTag/kubernetes-namespace": "ExampleNameSpace",
          "aws:RequestTag/kubernetes-service-account": "ExampleServiceAccountName"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-
role"
        }
      }
    }
  ]
}
```

La politica di cui sopra consente al ruolo `eks-pod-identity-primary-role` dell' AWS account 111122223333 con i relativi EKS Pod Identity Session Tag di assumere questo ruolo.

Se hai Disabilitato i tag di sessione nella tua EKS Pod Identity, EKS Pod Identity imposta anche `sts:ExternalId` con le informazioni sul cluster, sul namespace e sull'account di servizio di un pod quando assume un ruolo di destinazione.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
```

```
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "region/111122223333/cluster-name/namespace/service-
account-name"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-
role"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:TagSession"
    }
  ]
}
```

La policy di cui sopra aiuta a garantire che solo il cluster, il namespace e l'account di servizio previsti possano assumere il ruolo di destinazione.

Aggiorna la policy di autorizzazione per il ruolo EKS Pod Identity

In questo passaggio, aggiornerai la policy di autorizzazione del ruolo EKS Pod Identity associato al tuo cluster Amazon EKS aggiungendo l'ARN del ruolo IAM di destinazione come risorsa.

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster EKS.
3. Scegliere la scheda Accesso.
4. In Associazioni Pod Identity, seleziona il tuo ruolo EKS Pod Identity.
5. Scegli Autorizzazioni, scegli Aggiungi autorizzazioni, quindi Crea policy in linea.
6. Scegli JSON.
7. Nell'editor della policy, sostituisci il file JSON predefinito con la tua policy di autorizzazione. Sostituisci il valore segnaposto per il nome del ruolo e 222233334444 nell'ARN del ruolo IAM con il tuo ruolo IAM di destinazione. Esempio:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws:iam::222333334444:role/eks-pod-identity-aws-resources"
        }
    ]
}
```

Passaggio 2: associazione del ruolo IAM di destinazione a un account di servizio Kubernetes

In questo passaggio, creerai un'associazione tra il ruolo IAM di destinazione e l'account del servizio Kubernetes nel tuo cluster EKS.

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster a cui aggiungere l'associazione.
3. Scegli la scheda Accesso.
4. In Associazioni Pod Identity, scegli Crea.
5. Scegli il ruolo EKS Pod Identity nel ruolo IAM per i tuoi carichi di lavoro da assumere.
6. Scegli il ruolo IAM di destinazione nel ruolo IAM di destinazione che sarà assunto dal ruolo EKS Pod Identity.
7. Nel campo del namespace Kubernetes, inserisci il nome del namespace in cui desideri creare l'associazione (ad esempio, `my-app-namespace`). Ciò definisce dove risiede l'account del servizio.
8. Nel campo dell'account del servizio Kubernetes, inserisci il nome dell'account di servizio (ad esempio, `my-service-account`) che utilizzerà le credenziali IAM. Ciò collega il ruolo IAM all'account del servizio.
9. (Facoltativo) Seleziona Disabilita i tag di sessione per disabilitare i tag di sessione predefiniti che Pod Identity aggiunge automaticamente quando assume il ruolo.

10. (Facoltativo) Attiva Configure session policy per configurare una policy IAM per applicare restrizioni aggiuntive a questa associazione di identità Pod oltre alle autorizzazioni definite nella policy IAM allegata al ruolo IAM di Target.

> **Note**
>
> 1. Una policy di sessione può essere applicata solo quando l'impostazione Disabilita i tag di sessione è selezionata. 2. Se specifichi una policy di sessione, le restrizioni della policy si applicano alle autorizzazioni del ruolo IAM di Target e non al ruolo IAM associato a questa associazione Pod Identity.

11. Scegli Crea per creare l'associazione.

## Configura i pod per accedere ai AWS servizi con account di servizio

Se un Pod deve accedere ai AWS servizi, devi configurarlo per utilizzare un account di servizio Kubernetes. L'account di servizio deve essere associato a un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni per accedere ai servizi AWS .

- Un cluster esistente. Se non se ne possiede già uno, crearlo utilizzando una delle guide in Nozioni di base.
- Un account di servizio Kubernetes esistente e un'associazione EKS Pod Identity che associa l'account di servizio a un ruolo IAM. Al ruolo deve essere associata una policy IAM che contenga le autorizzazioni che desideri che i tuoi Pod abbiano per utilizzare i servizi. AWS Per ulteriori informazioni su come creare e configurare l'account e il ruolo del servizio, consulta the section called "Assegnazione di un ruolo IAM".
- La versione più recente della AWS CLI installata e configurata sul dispositivo o. AWS CloudShell È possibile verificare la versione corrente con `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl`

versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

1. Utilizza il seguente comando per creare un manifesto di implementazione per implementare un pod con cui confermare la configurazione. Sostituire i valori di esempio con i propri valori.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Implementa il file manifesto al cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Verifica che le variabili di ambiente richieste esistano per i pod.

   a. Visualizza i pod distribuiti con l'implementazione nella fase precedente.

```
kubectl get pods | grep my-app
```

   Di seguito viene riportato un output di esempio.

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

b. Verifica che il pod abbia il file di token dell'account di servizio montato.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

Di seguito viene riportato un output di esempio.

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. Verifica che i tuoi Pod possano interagire con i AWS servizi utilizzando le autorizzazioni che hai assegnato nella policy IAM associata al tuo ruolo.

> **Note**
>
> Quando un Pod utilizza AWS le credenziali di un ruolo IAM associato a un account di servizio, la AWS CLI o SDKs altro nei contenitori per quel Pod utilizza le credenziali fornite da quel ruolo. Se non si limita l'accesso alle credenziali fornite al ruolo IAM del nodo di Amazon EKS, il pod ha comunque accesso a tali credenziali. Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se il tuo pod non riesce a interagire con i servizi come previsto, completa i seguenti passaggi per confermare che tutto sia configurato correttamente.

a. Verifica che i tuoi Pod utilizzino una versione AWS SDK che supporti l'assunzione di un ruolo IAM tramite un'associazione EKS Pod Identity. Per ulteriori informazioni, consulta the section called "SDK supportati".

b. Conferma che l'implementazione stia utilizzando l'account del servizio.

```
kubectl describe deployment my-app | grep "Service Account"
```

Di seguito viene riportato un output di esempio.

```
Service Account:  my-service-account
```

## Concedi ai Pods l'accesso alle AWS risorse in base ai tag

Il controllo degli accessi basato su attributi (ABAC) concede diritti agli utenti attraverso policy che combinano gli attributi. EKS Pod Identity allega i tag alle credenziali temporanee di ciascun pod con attributi come nome cluster, namespace e nome account di servizio. Questi tag di sessione di ruolo consentono agli amministratori di creare un singolo ruolo che può funzionare su più account di servizio, consentendo l'accesso alle risorse in base ai tag corrispondenti. AWS Aggiungendo il supporto per i tag di sessione dei ruoli, puoi imporre limiti di sicurezza più rigorosi tra i cluster e i carichi di lavoro all'interno dei cluster, riutilizzando al contempo gli stessi ruoli IAM e le stesse policy IAM.

Policy esemplificativa con tag

Di seguito è riportato un esempio di policy IAM che concede a `s3:GetObject` autorizzazioni quando l'oggetto corrispondente è etichettato con il nome del cluster EKS.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

### Abilitare o disabilitare i tag di sessione

EKS Pod Identity aggiunge un set predefinito di tag di sessione quando assume il ruolo. Questi tag di sessione consentono agli amministratori di creare un singolo ruolo in grado di funzionare su più risorse, consentendo l'accesso alle AWS risorse in base ai tag corrispondenti.

#### Disattivare i tag di sessione

I tag di sessione sono abilitati automaticamente con EKS Pod Identity: non è richiesta alcuna azione da parte dell'utente. Per impostazione predefinita, EKS Pod Identity allega una serie di tag predefiniti alla sessione. Per fare riferimento a tali tag nelle politiche, utilizza la sintassi `${aws:PrincipalTag/` seguita dalla chiave del tag. Ad esempio, `${aws:PrincipalTag/kubernetes-namespace}`.

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

#### Disattiva i tag di sessione

AWS comprime i criteri di sessione in linea, i criteri ARNs gestiti e i tag di sessione in un formato binario compresso con un limite separato. Se ricevi un errore `PackedPolicyTooLarge` che indica che il formato binario compresso ha superato il limite di dimensione, puoi tentare ridurlo disattivando i tag di sessione aggiunti da EKS Pod Identity. Per disattivare questi tag di sessione, procedi nel seguente modo:

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster che desideri modificare.
3. Scegliere la scheda Accesso.
4. Nelle associazioni Pod Identity, scegli l'ID di associazione che desideri modificare in ID di associazione, quindi scegli Modifica.
5. In Tag di sessione, scegli Disattiva i tag di sessione.
6. Scegli Save changes (Salva modifiche).

Tag tra account

Tutti i tag di sessione aggiunti da EKS Pod Identity sono transitivi; le chiavi e i valori dei tag vengono passati a tutte le azioni `AssumeRole` utilizzate dai carichi di lavoro per cambiare ruolo in un altro account. È possibile utilizzare questi tag nelle policy di altri account per limitare l'accesso in scenari tra account. Per ulteriori informazioni, consulta Concatenamento dei ruoli con i tag di sessione nella Guida per l'utente IAM.

Tag personalizzati

EKS Pod Identity non può aggiungere tag personalizzati supplementari all'azione `AssumeRole` che esegue. Tuttavia, i tag applicati al ruolo IAM sono sempre disponibili nello stesso formato: `${aws:PrincipalTag/` seguito dalla chiave, ad esempio `${aws:PrincipalTag/MyCustomTag}`.

> **Note**
>
> I tag aggiunti alla sessione tramite la richiesta `sts:AssumeRole` hanno la precedenza in caso di conflitto. Ad esempio, supponiamo che:
>
> - Amazon EKS aggiunge una chiave `eks-cluster-name` e un valore `my-cluster` alla sessione quando EKS assume il ruolo di cliente e
> - Hai aggiunto anche un tag `eks-cluster-name` al ruolo IAM con valore `my-own-cluster`.
>
> In questo caso, il primo ha la precedenza e il valore del tag per `eks-cluster-name` sarà `my-cluster`.

## Usa l'identità del pod con AWS SDK

Utilizzo delle credenziali EKS Pod Identity

Per utilizzare le credenziali di un'associazione EKS Pod Identity, il codice può utilizzare qualsiasi AWS SDK per creare un client per un AWS servizio con un SDK e, per impostazione predefinita, l'SDK cerca in una catena di posizioni le credenziali di AWS Identity and Access Management da utilizzare. Le credenziali EKS Pod Identity verranno utilizzate se non si specifica un provider di credenziali quando si crea il client o si inizializza in altro modo l'SDK.

Questo avviene perché le associazioni EKS Pod Identity sono state aggiunte al provider di credenziali del container, che viene cercato in un passaggio nella catena di credenziali predefinita. Se i carichi di lavoro utilizzano attualmente credenziali che si trovano all'inizio della catena di credenziali, quest'ultime continuano a essere utilizzate anche se configuri un'associazione EKS Pod Identity per lo stesso carico di lavoro.

Per ulteriori informazioni sul funzionamento delle associazioni EKS Pod Identity, consulta the section called "Come funziona".

Quando si utilizza Learn how EKS Pod Identity concede ai pod l'accesso ai AWS servizi, i contenitori dei Pod devono utilizzare una versione AWS SDK che supporti l'assunzione di un ruolo IAM da parte di EKS Pod Identity Agent. Assicurati di utilizzare le seguenti versioni, o successive, per il tuo SDK: AWS

- Java (Versione 2) – 2.21.30
- Java — 1.12.746
- Vai a v1 – v1.47.11
- Vai a v2 – release-2023-11-14
- Python (Boto3) — 1.34.41
- Python (botocore) — 1.34.41
- AWS CLI — 1.30.0

  AWS CLI — 2.15.0
- JavaScript v2 — 2.1550.0
- JavaScript v3 — v3.458.0
- Kotlin – v1.0.1
- Ruby – 3.188.0
- Rust – release-2024-03-13
- C++ – 1.11.263
- .NET — 3.7.734.0
- PowerShell — 4.1.502
- PHP — 3.289.0

Per assicurarti che l'SDK utilizzato sia supportato, quando crei i container segui la procedura di installazione per il tuo SDK preferito disponibile in Strumenti per costruire su AWS.

Per un elenco di componenti aggiuntivi che supportano EKS Pod Identity, consulta the section called "Riferimento di supporto Pod Identity".

## Disabilita **IPv6** nell'EKS Pod Identity Agent

Console di gestione AWS

1. Per disabilitarlo `IPv6` nell'EKS Pod Identity Agent, aggiungi la seguente configurazione alle impostazioni di configurazione opzionali di EKS. Add-on
   a. Aprire la Console Amazon EKS.
   b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui configurare il componente aggiuntivo.
   c. Scegli la scheda Add-ons.
   d. Seleziona la casella nella parte superiore destra di quella del componente aggiuntivo relativo all'agente EKS Pod Identity, quindi scegli Modifica.
   e. Nella pagina Configura agente EKS Pod Identity:
      i. Seleziona la Versione da utilizzare. Si consiglia di mantenere la stessa versione rispetto al passaggio precedente e di aggiornare la versione e la configurazione con operazioni separate.
      ii. Scegliere Impostazioni di configurazione facoltative.
      iii. Inserisci la chiave JSON `"agent":` e il valore di un oggetto JSON annidato con una chiave `"additionalArgs":` in Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`. Di seguito è riportato l'esempio dell'abilitazione della policy di rete:

```
{
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
    }
}
```

      Questa configurazione imposta l'indirizzo `IPv4` come unico indirizzo utilizzato dall'agente.
   f. Per applicare la nuova configurazione sostituendo i pod dell'agente EKS Pod Identity, scegli Salva modifiche.

Amazon EKS applica le modifiche a EKS Add-ons utilizzando un'implementazione di Kubernetes `DaemonSet` for EKS Pod Identity Agent. Puoi tenere traccia dello stato dell'implementazione nella cronologia degli aggiornamenti del componente aggiuntivo in e con. Console di gestione AWS `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`

`kubectl rollout` dispone dei seguenti comandi:

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, iniziate dalla cronologia dell'implementazione ed eseguitelo `kubectl logs` su un pod EKS Pod Identity Agent per visualizzare i log di EKS Pod Identity Agent.

2. Se la nuova voce nella Cronologia degli aggiornamenti ha lo stato Riuscito, il rollout è stato completato e il componente aggiuntivo utilizza la nuova configurazione in tutti i pod dell'agente EKS Pod Identity.

AWS CLI

1. Per disabilitarlo `IPv6` nel EKS Pod Identity Agent, aggiungi la seguente configurazione ai valori di configurazione di EKS. Add-on

   Esegui il seguente comando AWS CLI. Sostituire `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che si sta utilizzando.

```
aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent \
    --resolve-conflicts PRESERVE --configuration-values '{"agent":{"additionalArgs":
 { "-b": "169.254.170.23"}}}'
```

Questa configurazione imposta l'indirizzo `IPv4` come unico indirizzo utilizzato dall'agente.

Amazon EKS applica le modifiche a EKS Add-ons utilizzando un'implementazione di Kubernetes DaemonSet for EKS Pod Identity Agent. Puoi tenere traccia dello stato dell'implementazione nella cronologia degli aggiornamenti del componente aggiuntivo in e con. Console di gestione AWS `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`

`kubectl rollout` dispone dei seguenti comandi:

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, iniziate dalla cronologia dell'implementazione ed eseguitelo `kubectl logs` su un pod EKS Pod Identity Agent per visualizzare i log di EKS Pod Identity Agent.

## Creazione di un ruolo IAM con la policy di fiducia richiesta da EKS Pod Identity

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
```

```
            ]
        }
    ]
}
```

**`sts:AssumeRole`**

EKS Pod Identity utilizza `AssumeRole` per assumere il ruolo IAM prima di passare le credenziali temporanee ai pod.

**`sts:TagSession`**

EKS Pod Identity utilizza `TagSession` per includere i tag di sessione nelle richieste a STS. AWS

Impostazione delle condizioni

È possibile utilizzare questi tag nelle chiavi di condizione nella policy di attendibilità per limitare gli account di servizio, i namespace e i cluster che possono utilizzare questo ruolo. Per l'elenco dei tag di richiesta aggiunti da Pod Identity, consulta the section called "Abilitare o disabilitare i tag di sessione".

Ad esempio, puoi limitare i pod che possono assumere un ruolo IAM di Pod Identity a uno specifico `ServiceAccount` e `Namespace` con la seguente politica di fiducia con l'aggiunta `Condition` di:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/kubernetes-namespace": [
                        "Namespace"
```

```
                    ],
                    "aws:RequestTag/kubernetes-service-account": [
                        "ServiceAccount"
                    ]
                }
            }
        }
    ]
}
```

Per un elenco di chiavi di condizione di Amazon EKS, consultare Condizioni per Amazon Elastic Kubernetes Service in Service Authorization Reference. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

# Manage compute resources by using nodes

Un nodo Kubernetes è una macchina che esegue applicazioni containerizzate. Ogni nodo include i seguenti componenti:

- Runtime del container: software responsabile dell'esecuzione dei container.
- kubelet: garantisce che i container siano integri e funzionanti nel proprio pod associato.
- kube-proxy: gestisce le regole della rete che consentono la comunicazione con i pod.

Per ulteriori informazioni, consulta Nodi nella documentazione Kubernetes.

Il cluster Amazon EKS può pianificare pod su qualsiasi combinazione di nodi gestiti dalla modalità automatica EKS, nodi autogestiti, gruppi di nodi gestiti da Amazon EKS, AWS Fargate e Amazon EKS Hybrid Nodes. Per ulteriori informazioni sui nodi implementati nel cluster, consulta the section called "Accesso alle risorse di cluster".

> **Note**
>
> A eccezione dei nodi ibridi, i nodi devono trovarsi nello stesso VPC delle sottoreti selezionate al momento della creazione del cluster, ma non necessariamente nelle stesse sottoreti.

## Confronto delle opzioni di calcolo

Nella tabella seguente vengono forniti diversi criteri per valutare quali opzioni sono più adatte alle proprie esigenze. I nodi autogestiti sono un'opzione alternativa che supporta tutti i criteri elencati, ma che richiede molta più manutenzione manuale. Per ulteriori informazioni, consulta the section called "Nodi autogestiti".

> **Note**
>
> Bottlerocket presenta alcune differenze specifiche rispetto alle informazioni generali contenute in questa tabella. Per ulteriori informazioni, consulta la documentazione di Bottlerocket su. GitHub

| Criteri | Gruppi di nodi gestiti EKS | Modalità automatica EKS | Amazon EKS Hybrid Nodes |
|---|---|---|---|
| Può essere implement ato in AWS Outposts | No | No | No |
| Si può implementare in una Zona locale AWS | Sì | No | No |
| Può eseguire container che richiedono Windows | Sì | No | No |
| Può eseguire container che richiedono Linux | Sì | Sì | Sì |
| Può eseguire carichi di lavoro che richiedono il chip Inferentia | Sì – Solo nodi Amazon Linux | Sì | No |
| Può eseguire carichi di lavoro che richiedono una GPU | Sì – Solo nodi Amazon Linux | Sì | Sì |
| Può eseguire carichi di lavoro che richiedono processori Arm | Sì | Sì | Sì |
| Può eseguire AWS Bottlerocket | Sì | Sì | No |
| I pod condivido no risorse di CPU, memoria, archiviaz | Sì | Sì | Sì |

| Criteri | Gruppi di nodi gestiti EKS | Modalità automatica EKS | Amazon EKS Hybrid Nodes |
|---|---|---|---|
| ione e rete con altri pod. | | | |
| Deve implementare e gestire le istanze Amazon EC2 | Sì | No - Ulteriori informazioni sulle istanze gestite di EC2 | Sì - Le macchine fisiche on-premises o virtuali sono autogestite con strumenti a scelta. |
| È necessario proteggere, mantenere e applicare patch al sistema operativo delle istanze Amazon EC2 | Sì | No | Sì - Il sistema operativo in esecuzione sulle macchine fisiche o virtuali è autogestito con strumenti a scelta. |
| Può fornire argomenti di bootstrap all'implementazione di un nodo, come argomenti kubelet aggiuntivi. | Sì - Utilizzando `eksctl` o un modello di avvio con un'AMI personalizzata. | No - Utilizzare `NodeClass` per configurare i nodi | Sì - È possibile personalizzare gli argomenti di bootstrap con nodeadm. Per informazioni, consulta the section called "nodeadm dei nodi ibridi". |
| Può assegnare indirizzi IP ai pod da un intervallo CIDR diverso rispetto all'indirizzo IP assegnato al nodo. | Sì: utilizzando un modello di avvio con un'AMI personalizzata. Per ulteriori informazioni, consulta the section called "Modelli di avvio". | No | Sì - Consultare the section called "Configurazione della CNI". |

| Criteri | Gruppi di nodi gestiti EKS | Modalità automatica EKS | Amazon EKS Hybrid Nodes |
|---|---|---|---|
| Puoi eseguire SSH nel nodo | Sì | No - Informazioni su come risolvere i problemi dei nodi | Sì |
| Puoi implementare un'AMI personalizzata nei nodi | Sì – Utilizzo di un modello di avvio | No | Sì |
| Può implementare un CNI personalizzato nei nodi | Sì – Utilizzando un modello di avvio con un'AMI personalizzata | No | Sì |

| Criteri | Gruppi di nodi gestiti EKS | Modalità automatica EKS | Amazon EKS Hybrid Nodes |
|---|---|---|---|
| É necessario aggiornare l'AMI del nodo per conto proprio | Sì - Se è stata implementata un'AMI ottimizzata per Amazon EKS, sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È possibile eseguire l'aggiornamento con un clic nella console. Se è stata implementata un'AMI personalizzata, non sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È necessario eseguire l'aggiornamento per conto proprio. | No | Sì - Il sistema operativo eseguito sulle macchine fisiche o virtuali è autogestito con strumenti a scelta. Per informazioni, consulta the section called "Preparazione del sistema operativo". |

| Criteri | Gruppi di nodi gestiti EKS | Modalità automatica EKS | Amazon EKS Hybrid Nodes |
|---|---|---|---|
| È necessario aggiornare la versione del nodo Kubernetes per conto proprio | Sì - Se è stata implementata un'AMI ottimizzata per Amazon EKS, sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È possibile eseguire l'aggiornamento con un clic nella console. Se è stata implementata un'AMI personalizzata, non sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È necessario eseguire l'aggiornamento per conto proprio. | No | Sì - Gli aggiornamenti dei nodi ibridi vengono gestiti con strumenti a scelta o con `nodeadm`. Per informazioni, consulta the section called "Aggiornamento dei nodi ibridi". |
| Può utilizzare lo spazio di archiviazione Amazon EBS con i pod | Sì | Sì, come funzionalità integrata. Informazioni su come creare una classe di archiviazione. | No |
| Può utilizzare lo spazio di archiviazione Amazon EFS con i pod | Sì | Sì | No |

| Criteri | Gruppi di nodi gestiti EKS | Modalità automatica EKS | Amazon EKS Hybrid Nodes |
|---|---|---|---|
| Può utilizzare lo storage Amazon S3 Files con Pods | Sì | Sì | No |
| Può usare Amazon FSx for Lustre storage con Pods | Sì | Sì | No |
| Può utilizzare Network Load Balancer per i servizi | Sì | Sì | Sì - È necessario utilizzare il tipo di destinazione `ip`. |
| I pod possono essere eseguiti in una sottorete pubblica | Sì | Sì | No - I pod vengono eseguiti in un ambiente on-premises. |
| Può assegnare diversi gruppi di sicurezza VPC a singoli pod | Sì – Solo nodi Linux | No | No |
| Può eseguire Kubernetes DaemonSets | Sì | Sì | Sì |
| Supporto per `HostPort` e `HostNetwork` nel manifesto pod | Sì | Sì | Sì |

| Criteri | Gruppi di nodi gestiti EKS | Modalità automatica EKS | Amazon EKS Hybrid Nodes |
|---|---|---|---|
| AWS Disponibilità regionale | Tutte le Regioni Amazon EKS supportate | Tutte le Regioni Amazon EKS supportate | Tutte le regioni supportate da Amazon EKS tranne le regioni AWS GovCloud (Stati Uniti) e le regioni della Cina. |
| Può eseguire container su host dedicati Amazon EC2 | Sì | No | No |
| Prezzi | Costo dell'istanza Amazon EC2 che esegue più pod. Per ulteriori informazioni, consulta Prezzi di Amazon EC2. | Quando la modalità automatica EKS è abilitata nel cluster, viene addebitata una tariffa separata per le istanze avviate utilizzan do la funzionalità di calcolo della modalità automatic a, in aggiunta ai costi standard delle istanze EC2. L'importo varia in base al tipo di istanza lanciata e alla AWS regione in cui si trova il cluster. Per ulteriori informazioni, consultare Prezzi di Amazon EKS. | Costo dei nodi ibridi in vCPU all'ora. Per ulteriori informazioni, consultare Prezzi di Amazon EKS. |

# Semplifica il ciclo di vita dei nodi con gruppi di nodi gestiti

I gruppi di nodi gestiti Amazon EKS automatizzano il provisioning la gestione del ciclo di vita dei nodi (istanze Amazon EC2) per i cluster Kubernetes Amazon EKS.

Con i gruppi di nodi gestiti Amazon EKS, non è necessario eseguire separatamente il provisioning o la registrazione delle istanze Amazon EC2 che forniscono capacità di calcolo per eseguire le applicazioni Kubernetes. È possibile creare, aggiornare o terminare automaticamente i nodi per il cluster con una singola operazione. Gli aggiornamenti e le interruzioni dei nodi svuotano automaticamente i nodi per garantire che le applicazioni rimangano disponibili.

Ogni nodo gestito viene assegnato come parte di un gruppo Amazon EC2 Auto Scaling gestito per l'utente da Amazon EKS. Ogni risorsa, incluse le istanze e i gruppi Auto Scaling, viene eseguita all'interno AWS del tuo account. Ogni gruppo di nodi viene eseguito su più zone di disponibilità definite.

I gruppi di nodi gestiti possono anche sfruttare opzionalmente la riparazione automatica, che monitora continuamente lo stato di integrità dei nodi. In pratica, reagisce automaticamente ai problemi rilevati e sostituisce i nodi quando possibile. Ciò aiuta la disponibilità complessiva del cluster con un intervento manuale minimo. Per ulteriori informazioni, consulta the section called "Salute e riparazione dei nodi".

Puoi aggiungere un gruppo di nodi gestiti a cluster nuovi o esistenti utilizzando la console Amazon EKS, la AWS CLI `eksctl` AWS , l'API o l'infrastruttura come strumenti di codice, tra cui. AWS CloudFormation I nodi avviati come parte di un gruppo di nodi gestiti vengono automaticamente taggati per l'individuazione automatica tramite Cluster Autoscaler di Kubernetes. É possibile utilizzare il gruppo di nodi per applicare le etichette Kubernetes ai nodi e aggiornarle in qualsiasi momento.

Non ci sono costi aggiuntivi per l'utilizzo dei gruppi di nodi gestiti di Amazon EKS, paghi solo per le AWS risorse fornite. Questi includono istanze Amazon EC2, volumi Amazon EBS, orari dei cluster Amazon EKS e qualsiasi altra infrastruttura. AWS Non sono previste tariffe minime né impegni anticipati.

Per iniziare a utilizzare un nuovo cluster Amazon EKS e un gruppo di nodi gestiti, vedi the section called "Crea cluster (console e CLI)".

Per aggiungere un gruppo di nodi gestiti a un cluster esistente, vedi the section called "Crea".

# Concetti sui gruppi di nodi gestiti

- I gruppi di nodi gestiti Amazon EKS creano e gestiscono le istanze Amazon EC2 per conto dell'utente.
- Ogni nodo gestito viene assegnato come parte di un gruppo Amazon EC2 Auto Scaling gestito per l'utente da Amazon EKS. Inoltre, ogni risorsa, incluse le istanze Amazon EC2 e i gruppi di Auto Scaling, viene eseguita all'interno del tuo account. AWS
- Amazon EKS sincronizza periodicamente la configurazione di scalabilità del gruppo di nodi gestiti in modo che corrisponda ai valori effettivi del gruppo Auto Scaling. Se un attore esterno come Cluster Autoscaler modifica la dimensione del gruppo Auto Scaling, `DescribeNodegroup` alla fine rifletterà tali modifiche. Quando si avvia un aggiornamento o un upgrade del gruppo di nodi senza modificare esplicitamente la configurazione di ridimensionamento, il flusso di lavoro utilizza i valori correnti del gruppo Auto Scaling anziché la configurazione di ridimensionamento memorizzata del gruppo di nodi. La configurazione di scalamento memorizzata ha la precedenza solo quando la includi esplicitamente in una richiesta. `UpdateNodegroupConfig`
- Il gruppo Auto Scaling di un gruppo di nodi gestiti si estende su ogni sottorete specificata al momento della creazione del gruppo.
- Amazon EKS applica tag alle risorse del gruppo di nodi gestiti in modo che siano configurate per utilizzare Cluster Autoscaler di Kubernetes.

> **Important**
>
> Se si esegue un'applicazione stateful in più zone di disponibilità supportate dai volumi Amazon EBS e che utilizza il Cluster Autoscaler di Kubernetes, è necessario configurare più gruppi di nodi, ognuno dei quali definito per una singola zona di disponibilità. Inoltre, è necessario abilitare la funzionalità `--balance-similar-node-groups`.

- È possibile utilizzare un modello di avvio personalizzato per un maggiore livello di flessibilità e personalizzazione durante l'implementazione dei nodi gestiti. Ad esempio, è possibile specificare argomenti `kubelet` aggiuntivi e utilizzare un'AMI personalizzata. Per ulteriori informazioni, consulta the section called "Modelli di avvio". Se non si utilizza un modello di avvio personalizzato quando si crea per la prima volta un gruppo di nodi gestiti, è disponibile un modello di avvio generato automaticamente. Non modificare manualmente questo modello generato automaticamente o si verificheranno errori.
- Amazon EKS segue il modello di responsabilità condivisa per i CVE e le patch di sicurezza sui gruppi di nodi gestiti. Quando i nodi gestiti eseguono un'AMI ottimizzata per Amazon EKS, Amazon

EKS è responsabile della creazione di versioni con patch dell'AMI quando vengono segnalati bug o problemi. Sei invece responsabile della distribuzione di queste versioni AMI con patch ai gruppi di nodi gestiti. Quando i nodi gestiti eseguono un'AMI personalizzata, sei responsabile della creazione di versioni con patch dell'AMI quando vengono segnalati bug o problemi, oltre che della distribuzione dell'AMI. Per ulteriori informazioni, consulta the section called "Aggiorna".

- I gruppi di nodi gestiti Amazon EKS possono essere avviati in sottoreti pubbliche e private. Se si avvia un gruppo di nodi gestiti in una sottorete pubblica in data 22 aprile 2020 o successiva, sarà necessario che la sottorete abbia `MapPublicIpOnLaunch` impostato su vero affinché le istanze possano partecipare correttamente a un cluster. Se la sottorete pubblica è stata creata utilizzando `eksctl` o i AWS CloudFormation modelli forniti da Amazon EKS a partire dal 26 marzo 2020, questa impostazione è già impostata su true. Se le sottoreti pubbliche sono state create prima del 26 marzo 2020 devi modificare manualmente l'impostazione. Per ulteriori informazioni, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete.
- Quando implementi un gruppo di nodi gestito in sottoreti private, devi assicurarti che possa accedere ad Amazon ECR per estrarre le immagini dei container. È possibile farlo collegando un gateway NAT alla tabella di routing della sottorete o aggiungendo i seguenti endpoint VPC AWS PrivateLink :
  - Interfaccia endpoint dell'API di Amazon ECR: `com.amazonaws.region-code.ecr.api`
  - Interfaccia endpoint dell'API del registro Docker di Amazon ECR: `com.amazonaws.region-code.ecr.dkr`
  - Endpoint del gateway Amazon S3: `com.amazonaws.region-code.s3`

  Per altri servizi ed endpoint di uso comune, consulta la sezione the section called "Cluster privati".
- I gruppi di nodi gestiti non possono essere distribuiti su AWS Outposts o in AWS Wavelength. È possibile creare gruppi di nodi gestiti in AWS Local Zones. Per ulteriori informazioni, consulta the section called " AWS Zone locali".
- È possibile creare più gruppi di nodi gestiti all'interno di un singolo cluster. Ad esempio, puoi creare un gruppo di nodi con l'AMI Linux standard ottimizzata per Amazon EKS per alcuni carichi di lavoro e un altro gruppo con la variante GPU per i carichi di lavoro che richiedono il supporto della GPU.
- Se il gruppo di nodi gestiti rileva un errore di controllo dello stato dell'istanza Amazon EC2, Amazon EKS restituirà un messaggio di errore che consente di diagnosticare il problema. Per ulteriori informazioni, consulta the section called "Codici di errore dei gruppi di nodi gestiti".
- Amazon EKS aggiunge etichette Kubernetes alle istanze del gruppo di nodi gestiti. Queste etichette fornite da Amazon EKS hanno il prefisso `eks.amazonaws.com`.

- Amazon EKS svuota automaticamente i nodi utilizzando l'API Kubernetes durante le interruzioni o gli aggiornamenti.
- I budget di interruzione del pod non vengono rispettati quando si termina un nodo con `AZRebalance` o si riduce il numero di nodi desiderato. Queste operazioni cercano di espellere pod dal nodo. Tuttavia, se trascorrono più di 15 minuti, il nodo viene terminato a prescindere dal fatto che tutti i pod sul nodo siano stati terminati. Per prolungare il periodo fino alla terminazione del nodo, aggiungi un hook del ciclo di vita al gruppo con scalabilità automatica. Per ulteriori informazioni, consulta Aggiungere un hook del ciclo di vita nella Guida per l'utente di Amazon EC2 Auto Scaling.
- Per eseguire correttamente il processo di scarico dopo aver ricevuto una notifica di interruzione spot o una notifica di ribilanciamento della capacità, `CapacityRebalance` deve essere impostato su `true`.
- L'aggiornamento di gruppi di nodi gestiti rispettano i budget di interruzione dei pod impostati per i pod. Per ulteriori informazioni, consulta the section called "Dettagli sull'aggiornamento del comportamento".
- I gruppi di nodi gestiti Amazon EKS non prevedono costi aggiuntivi. Paghi solo per le AWS risorse fornite.
- Se si desidera crittografare i volumi Amazon EBS per i nodi, è possibile implementare i nodi utilizzando un modello di avvio. Per implementare nodi gestiti con volumi Amazon EBS crittografati senza utilizzare un modello di avvio, crittografare tutti i nuovi volumi Amazon EBS creati nell'account. Per ulteriori informazioni, consulta la sezione Encryption by default nella Guida per l'utente di Amazon EC2.

## Tipi di capacità del gruppo di nodi gestiti

Quando crei un gruppo di nodi gestito, puoi scegliere il tipo di capacità On-Demand o Spot. Amazon EKS distribuisce un gruppo di nodi gestiti con un gruppo Amazon EC2 Auto Scaling che contiene On-Demand solo o solo istanze Spot di Amazon EC2. Puoi pianificare Pod per applicazioni con tolleranza ai guasti per gruppi di nodi gestiti da Spot e applicazioni intolleranti ai guasti per gruppi di nodi all' On-Demand interno di un singolo cluster Kubernetes. Per impostazione predefinita, un gruppo di nodi gestiti distribuisce istanze On-Demand Amazon EC2.

### On-Demand

Con On-Demand Instances, paghi la capacità di elaborazione al secondo, senza impegni a lungo termine.

Per impostazione predefinita, se non si specifica un tipo di capacità, al gruppo di nodi gestiti vengono fornite istanze. On-Demand Un gruppo di nodi gestiti configura un gruppo Amazon EC2 Auto Scaling per conto dell'utente con le seguenti impostazioni applicate:

- La strategia di allocazione per fornire On-Demand capacità è impostata su. `prioritized` I gruppi di nodi gestiti utilizzano l'ordine dei tipi di istanza passati nell'API per determinare quale tipo di istanza utilizzare per primo al momento del raggiungimento On-Demand della capacità. Ad esempio, è possibile specificare tre tipi di istanza nell'ordine seguente: `c5.large`, `c4.large`, e `c3.large`. Quando le On-Demand istanze vengono avviate, il gruppo di nodi gestiti raggiunge la On-Demand capacità iniziando con`c5.large`, poi e poi`c4.large`. `c3.large` Per maggiori informazioni, consultare Gruppo Amazon EC2 Auto Scaling nella Guida per l'utente di Amazon EC2 Auto Scaling.
- Amazon EKS aggiunge la seguente etichetta Kubernetes a tutti i nodi del gruppo di nodi gestito che specifica il tipo di capacità: `eks.amazonaws.com/capacityType: ON_DEMAND`. È possibile utilizzare questa etichetta per pianificare applicazioni con stato o intolleranti ai guasti sui nodi. On-Demand

## Spot

Le istanze Spot di Amazon EC2 sono una capacità di riserva di Amazon EC2 che offre forti sconti sui prezzi. On-Demand Quando EC2 necessita nuovamente della capacità in uso nelle istanze Spot Amazon EC2, questa essere interrotta con una notifica di interruzione di due minuti. Per ulteriori informazioni, consulta Spot Instances nella Guida per l'utente di Amazon EC2. É possibile configurare un gruppo di nodi gestito con Istanze spot Amazon EC2 per ottimizzare i costi per i nodi di calcolo in esecuzione nel tuo cluster Amazon EKS.

Per utilizzare istanze spot all'interno di un gruppo di nodi gestiti, creare un gruppo di nodi gestiti impostando il tipo di capacità come `spot`. Applicando le seguenti best practices Spot, un gruppo di nodi gestiti configura un gruppo Amazon EC2 Auto Scaling per conto dell'utente:

- Per garantire che il provisioning dei nodi Spot venga eseguito nei pool di capacità Spot ottimali, la strategia di allocazione è impostata su uno dei seguenti valori:
  - `price-capacity-optimized` (PCO): quando si creano nuovi gruppi di nodi in un cluster con Kubernetes versione `1.28` o successive, la strategia di allocazione è impostata su `price-capacity-optimized`. Tuttavia, la strategia di allocazione non viene modificata per i gruppi di nodi già creati con `capacity-optimized` prima che i gruppi di nodi gestiti da Amazon EKS inizino a supportare PCO.

- `capacity-optimized` (CO): quando si creano nuovi gruppi di nodi in un cluster con Kubernetes versione `1.27` o precedenti, la strategia di allocazione è impostata su `capacity-optimized`.

  Per aumentare il numero di pool di capacità Spot disponibili da cui allocare le capacità, configurare un gruppo di nodi gestiti per l'utilizzo di più tipi di istanza.

- Amazon EC2 Spot Capacity Rebalancing è abilitato in modo che Amazon EKS possa svuotare e bilanciare correttamente i nodi Spot per ridurre al minimo le interruzioni delle applicazioni quando un nodo Spot è a rischio elevato di interruzione. Per maggiori informazioni, consultare Rebalancing Capacity di Amazon EC2 Auto Scaling nella Guida per l'utente di Amazon EC2 Auto Scaling.
  - Quando un nodo Spot riceve un consiglio di ribilanciamento, Amazon EKS tenta automaticamente di avviare un nuovo nodo Spot sostitutivo.
  - Se un avviso di interruzione Spot di due minuti arriva prima che il nodo Spot sostitutivo si trovi in uno stato `Ready`, Amazon EKS inizia a svuotare il nodo Spot che ha ricevuto il suggerimento di ribilanciamento. Amazon EKS svuota il nodo nel modo migliore possibile. Di conseguenza, non c'è alcuna garanzia che Amazon EKS attenda che il nodo sostitutivo si unisca al cluster prima di svuotare il nodo esistente.
  - Quando un nodo Spot sostitutivo viene avviato ed è allo stato `Ready` su Kubernetes, Amazon EKS isola e svuota il nodo Spot che ha ricevuto il suggerimento di ribilanciamento. L'isolamento del nodo Spot assicura che il controller di servizio non invii nuove richieste a questo nodo Spot. Il nodo viene rimosso anche dall'elenco di nodi Spot sani e attivi. Lo svuotamento del nodo Spot assicura che i pod in esecuzione vengano espulsi in modo corretto.
- Amazon EKS aggiunge la seguente etichetta Kubernetes a tutti i nodi del gruppo di nodi gestito che specifica il tipo di capacità: `eks.amazonaws.com/capacityType: SPOT`. È possibile utilizzare questa etichetta per pianificare applicazioni fault tolerant sui nodi Spot.

> **Important**
>
> EC2 emette un avviso di interruzione Spot due minuti prima della chiusura dell'istanza Spot. Tuttavia, i Pod sui nodi Spot potrebbero non ricevere l'intera finestra di 2 minuti necessaria per un corretto spegnimento. Quando EC2 emette l'avviso, c'è un ritardo prima che Amazon EKS inizi a sfrattare i Pods. Gli sfratti avvengono in sequenza per proteggere il server dell'API Kubernetes, quindi durante più reclami Spot simultanei, alcuni Pod potrebbero ricevere avvisi di sfratto ritardato. I Pod possono essere chiusi forzatamente senza ricevere segnali di terminazione, in particolare su nodi ad alta densità di Pod, durante reclami simultanei o quando si utilizzano periodi di tolleranza di terminazione

prolungati. Per i carichi di lavoro Spot, consigliamo di progettare le applicazioni in modo da tollerare le interruzioni, utilizzando periodi di tolleranza di terminazione pari o inferiori a 30 secondi, evitando agganci PreStop di lunga durata e monitorando le metriche di sfratto dei Pod per comprendere i periodi di tolleranza effettivi nei cluster. Per i carichi di lavoro che richiedono una terminazione corretta e garantita, consigliamo invece di utilizzare la capacità. On-Demand

Nel decidere se implementare un gruppo di nodi con On-Demand o senza capacità Spot, è necessario considerare le seguenti condizioni:

- Le istanze Spot sono consigliate per applicazioni senza stato, fault tolerant e flessibili. Queste includono carichi di lavoro di formazione batch e machine learning, ETL di big data come Apache Spark, applicazioni di elaborazione delle code ed endpoint API senza stato. Poiché Spot è una capacità Amazon EC2 di riserva che può cambiare nel tempo, consigliamo di utilizzare la capacità Spot per carichi di lavoro con tolleranza di interruzione. Più specificamente, la capacità Spot è adatta per carichi di lavoro in grado di tollerare periodi in cui la capacità richiesta non è disponibile.
- Si consiglia di utilizzarlo On-Demand per applicazioni con intolleranza ai guasti. Ciò include strumenti di gestione dei cluster come strumenti di monitoraggio e operativi, implementazioni che richiedono `StatefulSets` e applicazioni con stato, come database.
- Per ottimizzare la disponibilità delle applicazioni durante l'utilizzo di istanze Spot, è consigliabile configurare un gruppo di nodi gestiti Spot per l'utilizzo di più tipi di istanza. Quando si utilizzano più tipi di istanza, si consiglia di applicare le seguenti regole:
  - All'interno di un gruppo di nodi gestiti, se utilizzi Cluster Autoscaler, ti consigliamo di utilizzare un set flessibile di tipi di istanza con la stessa quantità di vCPU e risorse di memoria. Questo per garantire che i nodi del cluster vengano dimensionati come previsto. Ad esempio, se sono necessarie quattro vCPU e otto memorie GiB, utilizzare `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` o altri tipi di istanza simili.
  - Per migliorare la disponibilità delle applicazioni, si consiglia di implementare più gruppi di nodi gestiti Spot. Per questo, ogni gruppo dovrebbe utilizzare un set flessibile di tipi di istanza con le stesse risorse vCPU e memoria. Ad esempio, se sono necessarie 4 vCPU e 8 memorie GiB, si consiglia di creare un gruppo di nodi gestiti con `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` o altri tipi di istanza simili; e un secondo gruppo di nodi gestiti con `m3.xlarge`, `m4.xlarge`, `m5.xlarge`, `m5d.xlarge`, `m5a.xlarge`, `m5n.xlarge` o altri tipi di istanza simili.

- Quando si implementa il gruppo di nodi con il tipo di capacità Spot che utilizza un modello di avvio personalizzato, utilizza l'API per passare più tipi di istanza. Non passare un singolo tipo di istanza tramite il modello di avvio. Per ulteriori informazioni sull'implementazione di un gruppo di nodi tramite un modello di avvio, vedere the section called "Modelli di avvio".

## Creare un gruppo di nodi gestiti per il cluster

In questo argomento viene descritto come avviare gruppi di nodi gestiti Amazon EKS per nodi che si registrano con il cluster Amazon EKS. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi.

Se si avvia un gruppo di nodi gestiti Amazon EKS per la prima volta, consigliamo invece di seguire una delle nostre guide in Nozioni di base. Queste guide forniscono procedure dettagliate per la creazione di un cluster Amazon EKS con nodi.

**Important**

- I nodi Amazon EKS sono istanze Amazon EC2 standard. La fatturazione viene effettuata in base ai normali prezzi Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.
- Non puoi creare nodi gestiti in una AWS regione in cui AWS Outposts o Wavelength AWS sono abilitati. È possibile, invece creare nodi autogestiti. Per ulteriori informazioni, consultare the section called "Amazon Linux", the section called "Windows" e the section called "Razzo per bottiglie". Puoi anche creare un gruppo di nodi Amazon Linux autogestito su un Outpost. Per ulteriori informazioni, consulta the section called "Nodi".
- Se non si specifica un ID AMI per il file `bootstrap.sh` incluso in Linux o Bottlerocket ottimizzato per Amazon EKS, i gruppi di nodi gestiti impongono un numero massimo sul valore di `maxPods`. Per le istanze con meno di 30 vCPU il numero massimo è `110`. Per le istanze con più di 30 vCPU il numero massimo passa a `250`. Questa applicazione ha la precedenza su altre configurazioni, tra cui. `maxPods` `maxPodsExpression` Per ulteriori informazioni su come `maxPods` viene determinato e su come personalizzarlo, vedere. the section called "Come viene determinato MaxPods"

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta the section called "Creazione di un cluster ".

- Un ruolo IAM esistente per i nodi da utilizzare. Per crearne uno, consulta the section called "Ruolo IAM del nodo". Se questo ruolo non prevede nessuna delle policy per VPC CNI, per i pod VPC CNI è necessario il ruolo separato riportato di seguito.
- (Facoltativo, ma consigliato) Il componente aggiuntivo plug-in CNI di Amazon VPC per Kubernetes configurato con il proprio ruolo IAM a cui è allegata la policy IAM necessaria. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
- Familiarità con le considerazioni presenti alla pagina Choose an optimal Amazon EC2 node instance type. A seconda del tipo di istanza scelto, potrebbero esserci ulteriori prerequisiti per il cluster e il VPC.
- Per aggiungere un gruppo di nodi gestiti da Windows, è prima necessario abilitare il supporto Windows per il cluster. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

È possibile creare un gruppo di nodi gestito con uno di questi due strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

## `eksctl`

Creare un gruppo di nodi gestito con eksctl

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

1. (Facoltativo) Se la policy IAM gestita AmazonEKS_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, si consiglia, invece, di assegnarla a un ruolo IAM associato all'account di servizio Kubernetes `aws-node`. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
2. Creare un gruppo di nodi gestiti con o senza utilizzare un modello di avvio personalizzato. La specifica manuale di un modello di avvio consente una maggiore personalizzazione di un gruppo

di nodi. Ad esempio, può consentire l'implementazione di un'AMI personalizzata o la presenza di argomenti sullo script `bootstrap.sh` in un'AMI ottimizzata per Amazon EKS. Per avere un elenco completo di tutte le opzioni e le impostazioni predefinite disponibili, inserisci il comando seguente.

```
eksctl create nodegroup --help
```

Nel comando seguente, sostituisci *my-cluster* con il nome del tuo cluster e sostituisci *my-mng* con il nome del gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti.

> **Important**
>
> Se non si utilizza un modello di avvio personalizzato durante la prima creazione di un gruppo di nodi gestiti, non utilizzarne uno per il gruppo di nodi in un secondo momento. Se non è stato specificato un modello di avvio personalizzato, il sistema genera automaticamente un modello di avvio che non è consigliabile modificare manualmente. La modifica manuale di questo modello di avvio generato in automatico potrebbe causare errori.

Senza un modello di avvio

`eksctl` crea un modello di avvio Amazon EC2 predefinito nell'account e implementa il gruppo di nodi utilizzando un modello di avvio creato in base alle opzioni specificate. Prima di specificare un valore per `--node-type`, consulta the section called "Tipi di istanza Amazon EC2".

Sostituisci *ami-family* con una parola chiave consentita. Per ulteriori informazioni, consulta Setting the node AMI Family (Impostazione della famiglia AMI dei nodi) nella documentazione di `eksctl`. Sostituisci *my-key* con il nome della coppia di chiavi Amazon EC2 o della chiave pubblica. Questa chiave viene utilizzata per eseguire il SSH nei nodi dopo il loro avvio.

> **Note**
>
> Per Windows, questo comando non abilita SSH. Associa, invece, la coppia di chiavi Amazon EC2 all'istanza e ti consente di eseguire il protocollo RDP nell'istanza.

Se non si dispone di una coppia di chiavi Amazon EC2, è possibile crearla nella Console di gestione AWS. Per informazioni su Linux, consultare Amazon EC2 key pairs and Linux instances nella Guida per l'utente di Amazon EC2. Per informazioni su Windows, consultare Amazon EC2 key pairs and Windows instances nella Guida per l'utente di Amazon EC2.

Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:

- Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
- Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se si desidera bloccare l'accesso del pod a IMDS, aggiungere l'opzione `--disable-pod-imds` al seguente comando.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key my-key
```

Le istanze possono opzionalmente assegnare un numero di indirizzi IP significativamente superiore ai pod, assegnare indirizzi IP ai pod da un intervallo CIDR diverso da quello dell'istanza ed essere distribuite in un cluster senza accesso a Internet. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP", the section called "Rete personalizzata", e the section called "Cluster privati" per ulteriori opzioni da aggiungere al comando precedente.

I gruppi di nodi gestiti calcolano e applicano un singolo valore per il numero massimo di Pod che possono essere eseguiti su ciascun nodo del gruppo di nodi, in base al tipo di istanza. Se si crea un gruppo di nodi con tipi di istanza diversi, il valore più piccolo calcolato in tutti i tipi di istanza viene applicato come numero massimo di pod che possono essere eseguiti su ogni tipo di istanza nel

gruppo di nodi. Per ulteriori informazioni su come viene calcolato questo valore, consultathe section called "Come viene determinato MaxPods".

Con un modello di lancio

Il modello di avvio deve esistere già e deve soddisfare i requisiti specificati nella pagina Launch template configuration basics. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:

- Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
- Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se si desidera bloccare l'accesso del pod a IMDS, specificare le impostazioni necessarie nel modello di avvio.

a. Copia i seguenti contenuti sul dispositivo. Sostituisci i valori di esempio e quindi esegui il comando modificato per creare il file. `eks-nodegroup.yaml` Diverse impostazioni specificate durante l'implementazione senza un modello di avvio vengono spostate nel modello di avvio. Se non si specifica una `version`, viene utilizzata la versione predefinita del modello.

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

Per un elenco completo delle impostazioni del file di configurazione `eksctl`, consulta Schema dei file config nella documentazione di `eksctl`. Le istanze possono opzionalmente assegnare un numero di indirizzi IP significativamente superiore ai pod, assegnare indirizzi IP ai pod da un

intervallo CIDR diverso da quello dell'istanza ed essere implementate in un cluster senza accesso a Internet in uscita. Per ulteriori informazioni, consultare the section called "Aumentare gli indirizzi IP", the section called "Rete personalizzata", e the section called "Cluster privati" per altre opzioni da aggiungere al file di configurazione.

Se non hai specificato un ID AMI nel modello di lancio, i gruppi di nodi gestiti calcolano e applicano un singolo valore per il numero massimo di Pod che possono essere eseguiti su ciascun nodo del gruppo di nodi, in base al tipo di istanza. Se si crea un gruppo di nodi con tipi di istanza diversi, il valore più piccolo calcolato in tutti i tipi di istanza viene applicato come numero massimo di pod che possono essere eseguiti su ogni tipo di istanza nel gruppo di nodi. Per ulteriori informazioni su come viene calcolato questo valore, consultathe section called "Come viene determinato MaxPods".

Se nel modello di avvio è stato specificato un ID AMI, specificare il numero massimo di pod che possono essere eseguiti su ciascun nodo del gruppo se si utilizza una rete personalizzata o se si vuole aumentare il numero di indirizzi IP assegnati all'istanza. Per ulteriori informazioni, consulta the section called "Come viene determinato MaxPods".

b. Implementare il gruppo di nodi mediante il comando seguente.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

## Console di gestione AWS

Creare un gruppo di nodi gestiti utilizzando la Console di gestione AWS

1. Attendi che lo stato del cluster risulti `ACTIVE`. Non è possibile creare un gruppo di nodi gestiti per un cluster non ancora `ACTIVE`.
2. Aprire la Console Amazon EKS.
3. Scegli il nome del cluster in cui desideri creare un gruppo di nodi gestiti.
4. Seleziona la scheda Compute (Calcolo).
5. Scegli Add node group (Aggiungi gruppo di nodi).
6. Nella pagina Configura il gruppo di nodi, compila i parametri di conseguenza e quindi scegli Successivo.
   - Nome: inserisci un nome univoco per il gruppo di nodi gestiti. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti.

- Ruolo IAM del nodo: scegli il ruolo dell'istanza del nodo da utilizzare con il gruppo di nodi. Per ulteriori informazioni, consulta the section called "Ruolo IAM del nodo".

> ⚠ Important
>
> - Non è possibile usare lo stesso ruolo utilizzato per creare i cluster.
> - Ti consigliamo di utilizzare un ruolo che non è attualmente utilizzato da nessun gruppo di nodi autogestito, a meno che tu non intenda utilizzarlo con un nuovo gruppo di nodi autogestito. Per ulteriori informazioni, consulta the section called "Elimina".

- Usa modello di avvio (Facoltativo): scegli se desideri utilizzare un modello di avvio esistente. Seleziona un Launch Template Name (Nome del modello di avvio). Quindi, seleziona una Launch template version (Versione del modello di avvio). Se non si seleziona una versione, Amazon EKS utilizza la versione predefinita del modello. I modelli di avvio permettono una maggiore personalizzazione del gruppo di nodi, ad esempio consentendo di implementare un'AMI personalizzata, assegnare un numero significativamente superiore di indirizzi IP ai pod, assegnare indirizzi IP ai pod da un intervallo CIDR diverso da quello dell'istanza e implementare i nodi in un cluster senza accesso a Internet in uscita. Per ulteriori informazioni, consultare the section called "Aumentare gli indirizzi IP", the section called "Rete personalizzata" e the section called "Cluster privati".

  Il modello di avvio deve soddisfare i requisiti descritti nella pagina Customize managed nodes with launch templates. Se non si utilizza il proprio modello, l'API di Amazon EKS crea un modello di avvio Amazon EC2 predefinito nell'account e implementa il gruppo di nodi utilizzando il modello di avvio predefinito.

  Se implementi ruoli IAM per gli account di servizio, assegni le autorizzazioni necessarie direttamente a ogni Pod che richiede l'accesso ai AWS servizi e nessun Pod nel tuo cluster richiede l'accesso a IMDS per altri motivi, come il recupero della AWS regione corrente, puoi anche disabilitare l'accesso a IMDS per i Pod che non utilizzano la rete host in un modello di lancio. Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).
- Etichette Kubernetes – (Facoltativo) È possibile scegliere di applicare etichette Kubernetes ai nodi del gruppo di nodi gestiti.
- Taint Kubernetes: (facoltativo) è possibile scegliere di applicare i taint Kubernetes ai nodi del gruppo di nodi gestiti. Nel menu Effetto sono disponibili le opzioni **`NoSchedule`** ,

**`NoExecute`** e **`PreferNoSchedule`** . Per ulteriori informazioni, consulta the section called "Impedire che i pod siano pianificati su nodi specifici".

- Tag: (facoltativo) è possibile scegliere di aggiungere tag al gruppo di nodi gestiti Amazon EKS. Questi tag non si propagano ad altre risorse del gruppo di nodi, ad esempio gruppi Auto Scaling o istanze. Per ulteriori informazioni, consulta the section called "Tagging delle risorse ".

7. Nella pagina Imposta configurazione di calcolo e dimensionamento, compila i parametri di conseguenza e quindi scegli Successivo.
   - Tipo di AMI: selezionare un tipo di AMI. Se si implementano istanze Arm, assicurarsi di valutare le considerazioni indicate alla pagina Amazon EKS optimized Arm Amazon Linux AMIs prima dell'implementazione.

     Se nella pagina precedente sono stati specificati un modello di avvio e un'AMI nel modello, non è possibile selezionare un valore. Viene visualizzato il valore del modello. L'AMI specificata nel modello deve soddisfare i requisiti indicati alla pagina Specifying an AMI.
   - Tipo di capacità: seleziona un tipo di capacità. Per ulteriori informazioni sulla scelta di un tipo di capacità, consulta the section called "Tipi di capacità del gruppo di nodi gestiti". Non è possibile combinare diversi tipi di capacità all'interno dello stesso gruppo di nodi. Se desideri utilizzare entrambi i tipi di capacità, crea gruppi di nodi separati, ognuno con i propri tipi di capacità e istanza. Per informazioni sul provisioning e il ridimensionamento dei nodi the section called "Gruppi di nodi gestiti e nodi autogestiti" di lavoro, vedere. GPU-accelerated
   - Tipi di istanza: per impostazione predefinita, viene specificato uno o più tipi di istanza. Per rimuovere un tipo di istanza predefinito, seleziona il X sul lato destro del tipo di istanza. Scegli il tipo di istanza da utilizzare nel gruppo di nodi gestiti. Per ulteriori informazioni, consulta the section called "Tipi di istanza Amazon EC2".

     Nella console viene visualizzato un insieme di tipi di istanza di uso comune. Se devi creare un gruppo di nodi gestito con un tipo di istanza che non viene visualizzato`eksctl`, usa la AWS CLI o un SDK per creare il gruppo di nodi. AWS CloudFormation Se nella pagina precedente è stato specificato un modello di avvio, non è possibile selezionare un valore perché il tipo di istanza deve essere specificato nel modello di avvio. Viene visualizzato il valore del modello di avvio. Se è stato selezionato Spot per Tipo capacità, al fine di migliorare la disponibilità è consigliabile specificare più tipi di istanza.
   - Dimensioni disco: inserire le dimensioni del disco (in GiB) da utilizzare per il volume root del nodo.

Se nella pagina precedente è stato specificato un modello di avvio, non è possibile selezionare un valore perché deve essere specificato nel modello di avvio.

- Dimensione desiderata: specifica il numero corrente di nodi che il gruppo di nodi gestiti deve mantenere all'avvio.

> **Note**
>
> Amazon EKS non aumenta o riduce orizzontalmente in automatico il gruppo di nodi. Tuttavia, è possibile configurare il Cluster Autoscaler di Kubernetes per eseguire questa operazione. Per ulteriori informazioni, consulta Cluster Autoscaler on. AWS

- Dimensione minima: specifica il numero minimo di nodi a cui il gruppo di nodi gestiti può essere ridotto.
- Dimensione massima: specifica il numero massimo di nodi a cui il gruppo di nodi gestiti può essere aumentato.
- Configurazione dell'aggiornamento dei gruppi di nodi: (facoltativo) puoi selezionare il numero o la percentuale di nodi da aggiornare in parallelo. Questi nodi non saranno disponibili durante l'aggiornamento. Per Numero massimo non disponibile, seleziona una delle seguenti opzioni e specifica un Valore:
  - Numero: seleziona e specifica il numero di nodi nel gruppo nodi che possono essere aggiornati in parallelo.
  - Percentuale: seleziona e specifica la percentuale di nodi nel gruppo di nodi che possono essere aggiornati in parallelo. Questa funzione è utile se disponi di un numero elevato di nodi nel gruppo di nodi.
- Configurazione di riparazione automatica del nodo: (facoltativo) se viene attivata la casella di controllo Abilita la riparazione automatica del nodo, Amazon EKS sostituirà automaticamente i nodi quando si verificano i problemi rilevati. Per ulteriori informazioni, consulta the section called "Salute e riparazione dei nodi".
- Configurazione piscina calda: (facoltativa) Se attivi la casella di controllo Abilita configurazione piscina calda, Amazon EKS creerà pool caldi sull'ASG. Per ulteriori informazioni, consulta the section called "Warm pool".

8. Nella pagina Specifica reti, compila i parametri opportunamente, quindi scegli Successivo.
   - Sottoreti: sceglii le sottoreti in cui avviare i nodi gestiti.

> **Important**
>
> Se si esegue un'applicazione stateful in più zone di disponibilità supportate dai volumi Amazon EBS e che utilizza il Cluster Autoscaler di Kubernetes, è necessario configurare più gruppi di nodi, ognuno dei quali definito per una singola zona di disponibilità. Inoltre, è necessario abilitare la funzionalità `--balance-similar-node-groups`.

> **Important**
>
> - Se si sceglie una sottorete pubblica e nel cluster è abilitato solo l'endpoint del server API pubblico, è necessario che la sottorete disponga di `MapPublicIP0nLaunch` impostato su `true` per consentire alle istanze di unirsi correttamente a un cluster. Se la sottorete è stata creata utilizzando `eksctl` o i AWS CloudFormation modelli forniti da Amazon EKS a partire dal 26 marzo 2020, allora questa impostazione è già impostata su. `true` Se le sottoreti sono state create con `eksctl` o i AWS CloudFormation modelli prima del 26 marzo 2020, devi modificare l'impostazione manualmente. Per ulteriori informazioni, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete.
> - Se si utilizza un modello di avvio e si specificano più interfacce di rete, Amazon EC2 non assegnerà automaticamente un indirizzo `IPv4` pubblico, anche se `MapPublicIp0nLaunch` è impostato su `true`. Affinché i nodi possano unirsi al cluster in questo scenario, è necessario abilitare l'endpoint del server API privato del cluster, oppure avviare i nodi in una sottorete privata con accesso Internet in uscita fornito attraverso un metodo alternativo, ad esempio un gateway NAT. Per ulteriori informazioni, consultare la pagina Amazon EC2 instance IP addressing della Guida per l'utente di Amazon EC2.

- Configurazione dell'accesso SSH ai nodi (facoltativo). L'abilitazione di SSH consente di connettersi alle istanze e raccogliere informazioni diagnostiche in caso di problemi. Consigliamo vivamente di abilitare l'accesso remoto quando crei un gruppo di nodi. Non è possibile abilitare l'accesso remoto dopo la creazione del gruppo di nodi.

  Se si sceglie di utilizzare un modello di avvio, questa opzione non viene visualizzata. Per abilitare l'accesso remoto ai nodi, specifica una coppia di chiavi nel modello di avvio e assicurati che la porta appropriata sia aperta ai nodi nei gruppi di sicurezza specificati nel modello di

avvio. Per ulteriori informazioni, consulta the section called "Utilizzo di gruppi di sicurezza personalizzati".

> **Note**
>
> Per Windows, questo comando non abilita SSH. Associa, invece, la coppia di chiavi Amazon EC2 all'istanza e ti consente di eseguire il protocollo RDP nell'istanza.

- Per Coppia di chiavi SSH (facoltativo), scegli una chiave SSH Amazon EC2 da utilizzare. Per informazioni su Linux, consultare Amazon EC2 key pairs and Linux instances nella Guida per l'utente di Amazon EC2. Per informazioni su Windows, consultare Amazon EC2 key pairs and Windows instances nella Guida per l'utente di Amazon EC2. Se si sceglie di utilizzare un modello di avvio, non è possibile selezionarne una. Quando viene fornita una chiave SSH di Amazon EC2 per i gruppi di nodi che utilizzano AMI Bottlerocket, viene attivato anche il container amministratore. Per ulteriori informazioni, consulta Container amministratore su GitHub.
- Per Autorizzazione di accesso remoto SSH da, se desideri limitare l'accesso a istanze specifiche, seleziona i gruppi di sicurezza associati a tali istanze. Se non si selezionano gruppi di sicurezza specifici, l'accesso SSH è consentito da qualsiasi indirizzo di Internet (`0.0.0.0/0`).

9. Nella pagina Rivedi e crea, controlla la configurazione del gruppo di nodi gestiti e scegli Crea.

   Se i nodi di lavoro non riescono a unirsi al cluster, consultare the section called "Impossibile aggiungere i nodi al cluster" nel capitolo dedicato alla risoluzione dei problemi.

10. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

11. (Solo nodi GPU) Se hai scelto un tipo di istanza GPU e un'AMI accelerata ottimizzata per Amazon EKS, devi applicare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo *vX.X.X* con la versione desiderata prima di eseguire il comando seguente. NVIDIA/k8s-device-plugin

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

## Installare i componenti aggiuntivi Kubernetes

Ora che si dispone di un cluster Amazon EKS funzionante con nodi, è possibile avviare l'installazione dei componenti aggiuntivi Kubernetes e l'implementazione di applicazioni al cluster. I seguenti argomenti della documentazione consentono di estendere la funzionalità del cluster.

- Il principale IAM in grado di creare il cluster è l'unico principale che può effettuare chiamate al server API Kubernetes tramite `kubectl` o la Console di gestione AWS. Se si desidera che altri principali IAM abbiano accesso al cluster, è necessario aggiungerli. Per ulteriori informazioni, consultare the section called "Accesso API Kubernetes" e the section called "Autorizzazioni richieste".
- Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
  - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
  - Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

  Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).
- Configurare il Cluster Autoscaler di Kubernetes per regolare automaticamente il numero di nodi nei gruppi di nodi.
- Implementare un'applicazione di esempio al cluster.
- Organizzare e monitorare le risorse del cluster con strumenti importanti per la gestione.

## Riduci la latenza per le applicazioni con tempi di avvio lunghi utilizzando pool caldi con gruppi di nodi gestiti

Quando le applicazioni hanno tempi di inizializzazione o avvio lunghi, gli eventi di scalabilità orizzontale possono causare ritardi: i nuovi nodi devono avviarsi completamente e unirsi al cluster prima di poter pianificare i Pod su di essi. Questa latenza può influire sulla disponibilità delle applicazioni durante picchi di traffico o eventi di scalabilità rapida. I pool caldi risolvono questo problema mantenendo un pool di istanze EC2 preinizializzate che hanno già completato il processo di avvio. Durante un evento di scalabilità orizzontale, le istanze passano dal pool caldo direttamente al cluster, evitando le lunghe fasi di inizializzazione e riducendo in modo significativo il tempo necessario alla disponibilità di nuova capacità. Per ulteriori informazioni, consulta Ridurre la latenza

per le applicazioni con tempi di avvio lunghi utilizzando pool caldi nella Guida per l'utente di Amazon EC2 Auto Scaling.

I gruppi di nodi gestiti di Amazon EKS supportano i pool caldi di Amazon EC2 Auto Scaling. Un pool caldo mantiene istanze EC2 preinizializzate insieme al gruppo Auto Scaling che può unirsi rapidamente al cluster durante gli eventi di scale-out. Le istanze nel pool caldo hanno già completato il processo di inizializzazione dell'avvio e possono essere mantenute nello stato, o. `Stopped` `Running` `Hibernated`

Amazon EKS gestisce i pool caldi durante l'intero ciclo di vita del gruppo di nodi utilizzando il ruolo `AWSServiceRoleForAmazonEKSNodegroup` collegato al servizio per creare, aggiornare ed eliminare le risorse del pool caldo.

Come funziona

Quando configuri un pool caldo, Amazon EKS crea un pool caldo EC2 Auto Scaling collegato al gruppo Auto Scaling del gruppo di nodi. Le istanze vengono avviate nel pool caldo, completano il processo di inizializzazione dell'avvio e rimangono nello stato configurato (`Running`, `Stopped` o) finché non sono necessarie. `Hibernated` Durante gli eventi di scalabilità orizzontale, le istanze passano dal pool caldo al gruppo Auto Scaling, completano il processo di inizializzazione di Amazon EKS per entrare a far parte del cluster e diventano disponibili per la pianificazione dei pod. Con il riutilizzo delle istanze abilitato, le istanze possono tornare al pool caldo durante gli eventi di scale-in.

> **Important**
>
> Configura sempre le pool calde tramite l'API Amazon EKS utilizzando `create-nodegroup` o`update-nodegroup-config`. Non modificare manualmente le impostazioni del pool caldo utilizzando l'API Auto Scaling di EC2, poiché ciò può causare conflitti con la gestione delle risorse di Amazon EKS.

Considerazioni

> **Important**
>
> Prima di configurare i pool caldi, esamina i prerequisiti e le limitazioni in Warm pools for Amazon EC2 Auto Scaling nella Amazon EC2 Auto Scaling User Guide. Non tutti i tipi di istanze o configurazioni sono supportati AMIs.

- Autorizzazioni IAM: il ruolo `AWSServiceRoleForAmazonEKSNodegroup` collegato al servizio (creato automaticamente con il primo gruppo di nodi gestito) include le necessarie autorizzazioni di gestione del pool caldo.
- Limitazioni AMI: le piscine calde non supportano la personalizzazione AMIs. È necessario utilizzare Amazon EKS ottimizzato AMIs.
- Limitazioni di Bottlerocket: se si utilizza Bottlerocket AMIs, lo stato del `Hibernated` pool non è supportato. Usa solo gli stati del `Stopped` pool`Running`. Inoltre, la `reuseOnScaleIn` funzionalità non è supportata con Bottlerocket AMIs.
- Supporto per l'ibernazione: lo stato del `Hibernated` pool è supportato solo su tipi di istanze specifici. Consulta i prerequisiti di ibernazione nella Guida per l'utente di Amazon EC2 per i tipi di istanze supportati.
- Impatto sui costi: la creazione di una piscina calda quando non è necessaria può comportare costi inutili.
- Pianificazione della capacità: dimensionate la piscina calda in base a modelli di scalabilità per bilanciare costi e disponibilità. Inizia con il 10-20% della capacità di picco prevista.
- Rete VPC: assicurate un numero sufficiente di indirizzi IP sia per le istanze del gruppo Auto Scaling che per le istanze warm pool.

### Configura le piscine calde

È possibile configurare i pool caldi durante la creazione di un nuovo gruppo di nodi gestiti o aggiornare un gruppo di nodi gestiti esistente per aggiungere il supporto per pool caldi.

### Parametri di configurazione

- abilitato — (booleano) Indica l'intenzione di collegare un pool caldo al gruppo di nodi gestito. Necessario per abilitare il supporto per piscine calde.
- maxGroupPreparedCapacità: (numero intero) Numero massimo di istanze totali tra pool caldo e gruppo Auto Scaling combinati.
- minSize — (numero intero) Numero minimo di istanze da mantenere nel pool caldo. Default: `0`.
- poolState — (stringa) Stato per le istanze del pool caldo. Default: `Stopped`.
- reuseOnScaleIn — (boolean) Indica se le istanze ritornano nel pool caldo durante gli eventi di scalabilità orizzontale anziché terminarli. Default: `false`. Non AMIs supportato con Bottlerocket.

Utilizzo della AWS CLI

È possibile configurare un pool caldo durante la creazione di un gruppo di nodi gestito o aggiungerne uno a un gruppo di nodi esistente.

Crea un gruppo di nodi con una piscina calda

```
aws eks create-nodegroup \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --node-role arn:aws:iam::111122223333:role/AmazonEKSNodeRole \
  --subnets subnet-12345678 subnet-87654321 \
  --region us-east-1 \
  --scaling-config minSize=2,maxSize=10,desiredSize=3 \
  --warm-pool-config
 enabled=true,maxGroupPreparedCapacity=8,minSize=2,poolState=Stopped,reuseOnScaleIn=true
```

Aggiungi un pool caldo a un gruppo di nodi esistente

```
aws eks update-nodegroup-config \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --region us-east-1 \
  --warm-pool-config
 enabled=true,maxGroupPreparedCapacity=8,minSize=2,poolState=Stopped,reuseOnScaleIn=true
```

Aggiornamento della configurazione

Aggiorna le impostazioni della piscina calda in qualsiasi momento utilizzandoupdate-nodegroup-config. Le istanze esistenti della piscina calda non vengono modificate immediatamente; le nuove impostazioni si applicano alle istanze che entrano nella piscina calda dopo l'aggiornamento.

```
aws eks update-nodegroup-config \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --region us-east-1 \
  --warm-pool-config
 enabled=true,maxGroupPreparedCapacity=10,minSize=3,poolState=Running,reuseOnScaleIn=true
```

Per disabilitare il pool caldo collegato al tuo gruppo di nodi, imposta: enabled=false

```
aws eks update-nodegroup-config \
```

```
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --region us-east-1 \
  --warm-pool-config enabled=false
```

Risorse aggiuntive

- Pool caldi per Amazon EC2 Auto Scaling nella Guida per l'utente di Amazon EC2 Auto Scaling
- Semplifica il ciclo di vita dei nodi con gruppi di nodi gestiti

## Aggiornamento del gruppo di nodi gestito per il cluster

Quando si avvia un aggiornamento di un gruppo di nodi gestiti, Amazon EKS aggiorna automaticamente i nodi, completando i passaggi elencati in Understand each phase of node updates. Se si utilizza un'AMI ottimizzata per Amazon EKS, Amazon EKS applica automaticamente le patch di sicurezza più recenti e gli aggiornamenti del sistema operativo ai nodi come parte dell'ultima versione di AMI.

Esistono diversi scenari in cui è utile aggiornare la versione o la configurazione del gruppo di nodi gestiti Amazon EKS:

- Hai aggiornato la versione di Kubernetes per il cluster Amazon EKS e vuoi aggiornare i nodi per utilizzare la stessa versione di Kubernetes.
- Una nuova versione dell'AMI è disponibile per il gruppo di nodi gestiti. Per ulteriori informazioni sulle versioni AMI, consultare le seguenti sezioni:
  - the section called "Ottenere informazioni sulla versione"
  - the section called "Portabottiglie"
  - the section called "Ottenere informazioni sulla versione"
- Per regolare il conteggio minimo, massimo o desiderato delle istanze nel gruppo di nodi gestiti.
- Per aggiungere o rimuovere le etichette Kubernetes dalle istanze del gruppo di nodi gestiti.
- Vuoi aggiungere o rimuovere AWS tag dal tuo gruppo di nodi gestito.
- È necessario implementare una nuova versione di un modello di avvio con modifiche alla configurazione, ad esempio un'AMI personalizzata aggiornata.
- Hai distribuito una versione `1.9.0` o successiva del componente aggiuntivo Amazon VPC CNI, abilitato il componente aggiuntivo per la delega dei prefissi e desideri che AWS nuove istanze Nitro

System in un gruppo di nodi supportino un numero significativamente maggiore di Pod. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP".

- Hai abilitato la delega del prefisso IP per i nodi Windows e desideri che nuove istanze di AWS Nitro System in un gruppo di nodi supportino un numero significativamente maggiore di Pod. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP".

Se esiste una versione dell'AMI più recente per la versione Kubernetes del gruppo di nodi gestiti rispetto a quella in esecuzione nel gruppo di nodi, è possibile aggiornare quest'ultima per utilizzare la nuova versione dell'AMI. Analogamente, se nel cluster è in esecuzione una versione di Kubernetes più recente rispetto al gruppo di nodi, è possibile aggiornare il gruppo di nodi per utilizzare la versione più recente dell'AMI che corrisponde alla versione di Kubernetes del cluster.

Quando un nodo in un gruppo di nodi gestiti è terminato a causa di un'operazione di dimensionamento o di un aggiornamento, i pod nel nodo sono svuotati per primi. Per ulteriori informazioni, consulta the section called "Dettagli sull'aggiornamento del comportamento".

### Aggiornare la versione di un gruppo di nodi

È possibile aggiornare la versione di un gruppo di nodi con uno dei seguenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

La versione a cui si aggiorna non può essere successiva alla versione del piano di controllo.

#### `eksctl`

Aggiornare un gruppo di nodi gestito utilizzando **`eksctl`**

Aggiorna un gruppo di nodi gestiti alla stessa versione AMI più recente di Kubernetes attualmente implementata nei nodi con il comando seguente. Sostituisci ogni *`example value`* con i valori in tuo possesso.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> Se stai aggiornando un gruppo di nodi implementato con un modello di avvio a una nuova versione del modello di avvio, aggiungi `--launch-template-version` *`version-number`* al precedente comando. Il modello di avvio deve soddisfare i requisiti descritti in Customize managed nodes with launch templates. Se il modello di avvio include un'AMI personalizzata, l'AMI deve soddisfare i requisiti in Specificare un'AMI. Quando si aggiorna il gruppo di nodi a una versione più recente del modello di avvio, tutti i nodi sono riciclati in modo da corrispondere alla nuova configurazione della versione del modello di avvio specificata.

Non è possibile aggiornare direttamente un gruppo di nodi implementato senza un modello di avvio a una nuova versione del modello di avvio. È invece necessario implementare un nuovo gruppo di nodi utilizzando il modello di avvio per aggiornare il gruppo di nodi a una nuova versione del modello di avvio.

È possibile aggiornare un gruppo di nodi alla stessa versione della versione Kubernetes del piano di controllo. Ad esempio, se hai un cluster che esegue Kubernetes `1.36`, puoi aggiornare i nodi che attualmente eseguono Kubernetes `1.35` alla versione `1.36` con il comando seguente.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.36
```

## Console di gestione AWS

Aggiornare un gruppo di nodi gestito utilizzando Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegliere il cluster che contiene il gruppo di nodi da aggiornare.
3. Se almeno un gruppo di nodi dispone di un aggiornamento disponibile, nella parte superiore della pagina viene visualizzata una casella di notifica dell'aggiornamento disponibile. Se selezioni la scheda Calcolo, visualizzerai Aggiorna ora nella colonna Versione rilascio AMI nella tabella Gruppi di nodi per il gruppo di nodi per cui è disponibile un aggiornamento. Per aggiornare il gruppo di nodi, scegli Update now (Aggiorna ora).

Non verrà visualizzata una notifica per i gruppi di nodi implementati con un'AMI personalizzata. Se i nodi vengono implementati con un'AMI personalizzata, completare la procedura seguente per implementare una nuova AMI personalizzata aggiornata.

a. Crea una nuova versione dell'AMI.

b. Creare una nuova versione del modello di avvio con il nuovo ID AMI.

c. Aggiornamento dei i nodi alla nuova versione del modello di avvio.

4. Nella finestra di dialogo Update node group version (Aggiorna la versione del gruppo di nodi), attiva o disattiva le seguenti opzioni:
   - Update node group version (Aggiorna la versione del gruppo di nodi): questa opzione non è disponibile se hai implementato un'AMI personalizzata o se l'AMI ottimizzata per Amazon EKS attualmente è disponibile nella versione più recente del cluster.
   - Change launch template version (Modifica la versione del modello di avvio): questa opzione non è disponibile se il gruppo di nodi è implementato senza un modello di avvio personalizzato. È possibile aggiornare la versione del modello di avvio solo per un gruppo di nodi implementato con un modello di avvio personalizzato. Seleziona la Launch template version (Versione del modello di avvio) a cui eseguire l'aggiornamento del gruppo di nodi. Se il gruppo di nodi è configurato con un'AMI personalizzata, anche la versione selezionata deve specificare un'AMI. Quando si esegue l'aggiornamento a una versione più recente del modello di avvio, tutti i nodi vengono riciclati in modo da corrispondere alla nuova configurazione della versione del modello di avvio specificata.
5. Per Update strategy (Strategia aggiornamento), seleziona una delle seguenti opzioni:
   - Aggiornamento in sequenza: questa opzione rispetta i budget di interruzione del pod per il cluster. Gli aggiornamenti non riescono se c'è un problema di budget che interrompe il pod, che fa sì che Amazon EKS non riesca a svuotare correttamente i pod in esecuzione su questo gruppo di nodi.
   - Forza aggiornamento: questa opzione non rispetta i budget per le interruzioni dei pod. Gli aggiornamenti sono eseguiti indipendentemente dai problemi di budget di interruzione del pod forzando il riavvio dei nodi.
6. Scegliere Aggiorna.

## Modificare la configurazione di un gruppo di nodi

È possibile modificare parte della configurazione di un gruppo di nodi gestiti.

1. Aprire la Console Amazon EKS.
2. Scegliere il cluster che contiene il gruppo di nodi da modificare.
3. Seleziona la scheda Compute (Calcolo).
4. Seleziona il gruppo di nodi da modificare, e scegli Edit (Modifica).
5. (Facoltativo) Nella pagina Edit node group (Modifica gruppo di nodi), effettua le seguenti operazioni:
   a. Modifica la configurazione di scalabilità del gruppo di nodi.
      - Dimensione desiderata: specifica il numero corrente di nodi che il gruppo di nodi gestiti deve mantenere.
      - Dimensione minima: specifica il numero minimo di nodi a cui il gruppo di nodi gestiti può essere ridotto.
      - Dimensione massima: specifica il numero massimo di nodi a cui il gruppo di nodi gestiti può essere aumentato orizzontalmente. Per il numero massimo di nodi supportati in un gruppo di nodi, vedere the section called "Service Quotas".
   b. (Facoltativo) Aggiungi o rimuovi etichette Kubernetes ai nodi nel gruppo di nodi. Le etichette mostrate qui sono solo le etichette applicate con Amazon EKS. Altre etichette, non mostrate qui, possono essere presenti sui nodi.
   c. (Facoltativo) Aggiungi o rimuovi taint Kubernetes ai nodi nel gruppo di nodi. I taint aggiunti possono avere l'effetto di **`NoSchedule`**, **`NoExecute`**, oppure **`PreferNoSchedule`**. Per ulteriori informazioni, consulta the section called "Impedire che i pod siano pianificati su nodi specifici".
   d. (Facoltativo) Aggiungi o rimuovi Tag dalla risorsa del gruppo di nodi. Questi tag vengono applicati solo al gruppo di nodi Amazon EKS. I tag dei gruppi di nodi non si propagano ad altre risorse come istanze o le sottoreti di Amazon EC2 nel gruppo di nodi.
   e. (Facoltativo) Modifica la Configurazione dell'aggiornamento del gruppo di nodi. Seleziona un'opzione tra Numero o Percentuale.
      - Numero: seleziona e specifica il numero di nodi nel gruppo nodi che possono essere aggiornati in parallelo. Questi nodi non saranno disponibili durante l'aggiornamento.
      - Percentuale: selezionare e specificare la percentuale di nodi nel gruppo di nodi che possono essere aggiornati in parallelo. Questi nodi non saranno disponibili durante l'aggiornamento. Questa funzione è utile se si dispone di diversi nodi nel gruppo di nodi.
   f. Al termine della modifica, scegli Salva modifiche.

> **Important**
>
> Quando si aggiorna la configurazione del gruppo di nodi, la modifica di `NodegroupScalingConfig` non rispetta i budget di interruzione pod (PDB). A differenza del processo di aggiornamento del gruppo di nodi (che drena i nodi e rispetta i PDB durante la fase di aggiornamento), l'aggiornamento della configurazione di scalabilità causa la chiusura immediata dei nodi tramite una chiamata di riduzione verticale del gruppo Auto Scaling (ASG). Ciò avviene senza considerare i PDB, indipendentemente dalla dimensione di destinazione a cui stai ridimensionando verticalmente. Ciò significa che quando riduci `desiredSize` di un gruppo di nodi gestiti da Amazon EKS, i pod sono espulsi non appena i nodi sono terminati, senza onorare alcun PDB.

## Esegui il rollback di una versione gestita di un gruppo di nodi

Se devi ripristinare il piano di controllo del cluster a una versione precedente di Kubernetes, devi prima ripristinare i gruppi di nodi gestiti per garantire la conformità alla politica di inclinazione della versione di Kubernetes. I nodi di lavoro non possono eseguire una versione più recente del piano di controllo.

Puoi ripristinare un gruppo di nodi gestito utilizzando l'`UpdateNodegroupVersion`API con la versione precedente di Kubernetes:

```
aws eks update-nodegroup-version \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --kubernetes-version 1.30 \
  --region us-west-2
```

Il rollback del gruppo di nodi rispetta le impostazioni di aggiornamento configurate (`maxUnavailableomaxUnavailablePercentage`) e la strategia di aggiornamento (Rolling o Force), proprio come un aggiornamento.

Per ulteriori informazioni sull'intero processo di rollback del cluster, vedere. the section called "Ripristina la versione di Kubernetes"

# Comprendi ogni fase degli aggiornamenti dei nodi

La strategia di aggiornamento del nodo (worker) gestito di Amazon EKS ha quattro fasi diverse descritte nelle sezioni seguenti.

## Fase di configurazione

La fase di configurazione prevede i seguenti passaggi:

1. Crea una nuova versione del modello di avvio Amazon EC2 per il gruppo Auto Scaling associato al gruppo di nodi. La nuova versione del modello di avvio utilizza l'AMI di destinazione o una versione del modello di avvio personalizzata per l'aggiornamento.
2. Il gruppo Auto Scaling viene aggiornato per utilizzare la versione più recente del modello di avvio.
3. Determina la quantità massima di nodi da aggiornare in parallelo utilizzando la proprietà `updateConfig` per il gruppo di nodi. Il massimo non disponibile ha una quota di 100 nodi. Il valore di default è un nodo. Per ulteriori informazioni, consulta la proprietà updateConfig nella Documentazione di riferimento delle API di Amazon EKS.

## Fase di scalabilità

Quando si aggiornano i nodi in un gruppo di nodi gestiti, i nodi aggiornati vengono avviati nella stessa zona di disponibilità di quelli in fase di aggiornamento. Per garantire questo posizionamento, utilizziamo il ribilanciamento della zona di disponibilità di Amazon EC2. Per ulteriori informazioni, consulta Availability Zone Rebalancing nella Guida per l'utente di Amazon EC2 Auto Scaling. Per soddisfare questo requisito, è possibile avviare fino a due istanze per zona di disponibilità nel gruppo di nodi gestiti.

La fase di scalabilità prevede i seguenti passaggi:

1. Incrementa la dimensione massima e la dimensione desiderata del gruppo Auto Scaling in base alla scelta maggiore tra:
   - Fino a due volte il numero di zone di disponibilità in cui è stato implementato il gruppo Auto Scaling.
   - Il massimo non disponibile per l'aggiornamento.

   Ad esempio, se il gruppo di nodi ha cinque zone di disponibilità e `maxUnavailable` è uno, il processo di aggiornamento può avviare al massimo 10 nodi. Tuttavia, quando `maxUnavailable` è 20 (o comunque superiore a 10), il processo avvia 20 nuovi nodi.

2. Dopo aver dimensionato il gruppo Auto Scaling, verifica se i nodi che utilizzano la configurazione più recente sono presenti nel gruppo di nodi. Questo passaggio ha esito positivo solo quando soddisfa i seguenti criteri:
   - Almeno un nuovo nodo viene avviato in ogni zona di disponibilità in cui esiste il nodo.
   - Ogni nuovo nodo deve essere nello stato `Ready`.
   - I nuovi nodi devono avere etichette applicate da Amazon EKS.

     Queste sono le etichette applicate da Amazon EKS sui nodi (worker) di un normale gruppo di nodi:

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`

     Queste sono le etichette applicate da Amazon EKS sui nodi (worker) in un modello di avvio personalizzato o un gruppo di nodi AMI:

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`
     - `eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId`
     - `eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion`

     > **Note**
     >
     > Quando viene avviato un aggiornamento o un upgrade senza modifiche alla configurazione di ridimensionamento, il flusso di lavoro utilizza i valori del gruppo Auto Scaling in tempo reale come punto di partenza, non la configurazione di ridimensionamento memorizzata del gruppo di nodi. Per ulteriori informazioni, consulta the section called "Concetti sui gruppi di nodi gestiti".

3. Contrassegna i nodi come non programmabili per evitare di programmare nuovi pod. Inoltre, etichetta i nodi con `node.kubernetes.io/exclude-from-external-load-balancers=true` per rimuovere i vecchi nodi dai sistemi di bilanciatore del carico prima di terminare i nodi.

Di seguito sono riportati i motivi noti che portano a un errore `NodeCreationFailure` in questa fase:

Capacità insufficiente nella zona di disponibilità

Esiste la possibilità che la zona di disponibilità non abbia capacità disponibile per i tipi di istanza richiesti. Si consiglia di configurare più tipi di istanze durante la creazione di un gruppo di nodi gestiti.

Limiti delle istanze EC2 dell'account

Potrebbe essere necessario aumentare il numero di istanze Amazon EC2 che il l'account può eseguire contemporaneamente utilizzando Service Quotas. Per ulteriori informazioni, consulta Service Quotas di EC2 nella Guida per l'utente di Amazon Elastic Compute Cloud per le istanze Linux.

Dati utente personalizzati

I dati utente personalizzati possono talvolta interrompere il processo di bootstrap. Questo scenario può portare al mancato avvio di `kubelet` sul nodo o sui nodi che non ricevono le etichette Amazon EKS previste. Per ulteriori informazioni, consulta the section called "Specifica di un'AMI".

Qualsiasi modifica che renda un nodo non integro o non pronto

La pressione del disco del nodo, la pressione della memoria e condizioni simili possono far sì che un nodo non assuma lo stato `Ready`.

Ogni nodo deve avviarsi entro 15 minuti

Se un nodo impiega più di 15 minuti per avviarsi e unirsi al cluster, si verificherà il timeout dell'aggiornamento. Questo è il runtime totale per il bootstrap di un nuovo nodo misurato dal momento in cui è necessario un nuovo nodo al momento in cui si unisce al cluster. Quando si aggiorna un gruppo di nodi gestito, il contatore temporale si avvia non appena la dimensione del gruppo Auto Scaling aumenta.

## Fase di aggiornamento

La fase di aggiornamento si comporta in due modi diversi, a seconda della strategia di aggiornamento. Esistono due strategie di aggiornamento: predefinita e minimale.

Consigliamo una strategia predefinita nella maggior parte degli scenari. Infatti, questa crea nuovi nodi prima di terminare quelli vecchi, in modo da mantenere la capacità disponibile durante la fase di aggiornamento. La strategia minimale è utile in scenari in cui si è limitati per quanto riguarda risorse o costi, ad esempio con acceleratori hardware come le GPU. Termina i vecchi nodi prima di creare quelli nuovi, in modo che la capacità totale non aumenti mai oltre la quantità configurata.

La strategia di aggiornamento predefinita prevede i seguenti passaggi:

1. Aumenta la quantità di nodi (numero desiderato) nel gruppo Auto Scaling, facendo sì che il gruppo di nodi crei nodi aggiuntivi.
2. Seleziona casualmente un nodo che deve essere aggiornato, fino al massimo non disponibile configurato per il gruppo di nodi.
3. Blocca il vecchio nodo non appena un nuovo nodo raggiunge lo stato Ready, impedendo al controller di servizio di inviare nuove richieste a quel nodo.
4. Svuota i pod dal nodo. Se i pod non lasciano il nodo entro 15 minuti e non c'è un flag di forzatura, la fase di aggiornamento non riesce con errore `PodEvictionFailure`. Per questo scenario, puoi applicare il flag di forzatura con la richiesta `update-nodegroup-version` di eliminare i pod.
5. Attende 60 secondi dopo aver sfrattato ogni Pod. Quindi invia una richiesta di terminazione all'Auto Scaling Group e rimuove il nodo dall'elenco dei nodi attivi.
6. Ripete i passaggi di aggiornamento precedenti fino a quando non vi sono nodi nel gruppo implementato con la versione precedente del modello di avvio.

La strategia di aggiornamento minimale prevede i seguenti passaggi:

1. All'inizio isola tutti i nodi del gruppo di nodi, in modo che il controller di servizio non invii nuove richieste a questi nodi.
2. Seleziona casualmente un nodo che deve essere aggiornato, fino al massimo non disponibile configurato per il gruppo di nodi.
3. Drena i Pod dai nodi selezionati. Se i pod non lasciano il nodo entro 15 minuti e non c'è un flag di forzatura, la fase di aggiornamento non riesce con errore `PodEvictionFailure`. Per questo scenario, puoi applicare il flag di forzatura con la richiesta `update-nodegroup-version` di eliminare i pod.
4. Dopo che ogni pod è stato rimosso e ha atteso 60 secondi, invia una richiesta di interruzione al gruppo Auto Scaling per i nodi selezionati. Il gruppo Auto Scaling crea nuovi nodi (lo stesso numero di nodi selezionati) per sostituire la capacità mancante.
5. Ripete i passaggi di aggiornamento precedenti fino a quando non vi sono nodi nel gruppo implementato con la versione precedente del modello di avvio.

### Errori **`PodEvictionFailure`** durante la fase di aggiornamento

Di seguito sono riportati i motivi noti che portano a un errore `PodEvictionFailure` in questa fase:

PDB aggressivo

Sul pod è definito un PDB aggressivo oppure sono presenti più PDB che puntano allo stesso pod.

Implementazione che tollera tutte le contaminazioni

Una volta espulso ogni pod, il nodo dovrebbe essere vuoto in quanto oggetto di taint nei passaggi precedenti. Tuttavia, se l'implementazione tollera ogni taint, è più probabile che il nodo non sia vuoto, causando un errore di espulsione dal pod.

### Fase di dimensionamento

La fase di dimensionamento diminuisce di uno la dimensione massima del gruppo Auto Scaling e la dimensione desiderata per tornare ai valori prima dell'avvio dell'aggiornamento.

Se il flusso di lavoro di aggiornamento determina che il Cluster Autoscaler stia dimensionando il gruppo di nodi durante la fase di dimensionamento del flusso di lavoro, esce immediatamente senza riportare il gruppo di nodi alle dimensioni originali.

**Note**

```
If your node group has a warm pool enabled, warm pool instances are drained
 before the scale-up operation begins. This is because warm pool instances have
 not been updated to the new launch template configuration – during the scale-
up phase, they would be pulled into the Auto Scaling Group instead of launching
 new instances with the updated configuration, which would break the upgrade
 process. Draining the warm pool ensures that only new instances with the
 updated configuration are launched. Once the scale-down operation completes,
 the warm pool is restored, and the new instances in the warm pool are launched
 with the updated launch template configuration.
```

Per ulteriori informazioni sui warm pool, consulta the section called "Warm pool".

## Personalizzazione dei nodi gestiti con modelli di avvio

Per il massimo livello di personalizzazione, puoi distribuire nodi gestiti con il tuo modello di lancio in base ai passaggi di questa pagina. L'utilizzo di un modello di avvio consente funzionalità come fornire argomenti di bootstrap durante la distribuzione di un nodo (ad esempio, argomenti kubelet aggiuntivi), assegnare indirizzi IP ai Pod da un blocco CIDR diverso dall'indirizzo IP assegnato al

nodo, distribuire la propria AMI personalizzata ai nodi o distribuire il proprio CNI personalizzato sui nodi.

Quando fornisci il tuo modello di avvio alla prima creazione di un gruppo di nodi gestito, in seguito avrai anche una maggiore flessibilità. Dopo aver implementato un gruppo di nodi gestiti con un modello di avvio personalizzato, potrai aggiornarlo in maniera iterativa con una versione diversa dello stesso modello. Quando si aggiorna il gruppo di nodi a una versione diversa del modello di avvio, tutti i nodi del gruppo vengono riciclati in modo da corrispondere alla nuova configurazione della versione del modello di avvio specificata.

I gruppi di nodi gestiti vengono sempre implementati con un modello di avvio da utilizzare con gruppo di Amazon EC2 Auto Scaling. Quando non è fornito un modello di avvio, l'API Amazon EKS ne crea automaticamente uno con i valori predefiniti nel tuo account. Tuttavia, consigliamo di non modificare i modelli di avvio generati automaticamente. Inoltre, i gruppi di nodi esistenti che non utilizzano un modello di avvio personalizzato non possono essere aggiornati direttamente. A tale scopo, sarà invece necessario creare un nuovo gruppo di nodi con un modello di avvio personalizzato.

## Informazioni di base sulla configurazione del modello di avvio

Puoi creare un modello di lancio di Amazon EC2 Auto Scaling con la Console di gestione AWS CLI AWS o un SDK. AWS Per ulteriori informazioni, consulta Creazione di un modello di avvio per un gruppo con scalabilità automatica nella Guida per l'utente di Amazon EC2 Auto Scaling. Alcune impostazioni di un modello di avvio sono simili a quelle utilizzate per la configurazione dei nodi gestiti. Quando si implementa o si aggiorna un gruppo di nodi con un modello di avvio, è necessario specificare alcune impostazioni nella configurazione del gruppo di nodi o nel modello di avvio. Non specificare un'impostazione in entrambe le posizioni. Se un'impostazione è impostata dove non dovrebbe, operazioni come la creazione o l'aggiornamento di un gruppo di nodi non hanno esito positivo.

Nella tabella seguente sono elencate le impostazioni vietate in un modello di avvio. Elenca inoltre impostazioni simili, se disponibili, che sono necessarie per la configurazione del gruppo di nodi gestiti. Le impostazioni elencate sono quelle visualizzate nella console. Potrebbero avere nomi simili ma diversi nella AWS CLI e nell'SDK.

| Modello di avvio – Vietato | Configurazione del gruppo di nodi Amazon EKS |
| --- | --- |
| Sottorete in Interfacce di rete (Aggiungi interfaccia di rete) | Sottoreti in Configurazione di rete del gruppo di nodi nella pagina Specifica reti |

| Modello di avvio – Vietato | Configurazione del gruppo di nodi Amazon EKS |
| --- | --- |
| Profilo dell'istanza IAM in Dettagli avanzati | Ruolo IAM del nodo in Configurazione del gruppo di nodi nella pagina Configura gruppo di nodi |
| Comportamento di arresto e Interrompi - Iberna comportamento in Dettagli avanzati. Mantieni default Non includere nell'impostazione del modello di avvio nel modello di avvio per entrambe le impostazioni. | Nessun equivalente. Amazon EKS deve controllare il ciclo di vita dell'istanza, non il gruppo Auto Scaling. |

Nella tabella seguente sono elencate le impostazioni vietate nella configurazione di un gruppo di nodi gestiti. Elenca inoltre impostazioni simili, se disponibili, richieste in un modello di avvio. Le impostazioni elencate sono quelle visualizzate nella console. Potrebbero avere nomi simili nella AWS CLI e nell'SDK.

| Configurazione del gruppo di nodi Amazon EKS – Vietata | Modello di avvio |
| --- | --- |
| (Solo se è stata specificata un'AMI personalizzata in un modello di avvio) Tipo di AMI in Configurazione di calcolo del gruppo di nodi sulla pagina Imposta configurazione di calcolo e dimensionamento: la console mostra il messaggio Specificato nel modello di lancio e l'ID AMI specificato.<br><br>Se nel modello di avvio non è stato specificato Applicazione e immagini OS (Amazon Machine Image), è possibile selezionare un'AMI nella configurazione del gruppo di nodi. | Applicazione e immagini OS (Amazon Machine Image) in Contenuti del modello di avvio: è necessario specificare un ID se soddisfi uno dei seguenti requisiti:<br><br>• É in esecuzione un'AMI personalizzata. Se si specifica un'AMI che non soddisfa i requisiti elencati in Specificare un'AMI, l'impleme ntazione del gruppo di nodi non avrà esito positivo.<br>• Necessità di fornire i dati utente per passare gli argomenti al file `bootstrap.sh` incluso nell'AMI ottimizzata per Amazon EKS. È possibile abilitare le istanze in modo che assegnino un numero significativamente più elevato di indirizzi IP ai pod, assegnare |

| Configurazione del gruppo di nodi Amazon EKS – Vietata | Modello di avvio |
| --- | --- |
| | indirizzi IP ai pod da un intervallo CIDR diverso da quello dell'istanza o implementare un cluster privato senza accesso a Internet in uscita. Per ulteriori informazioni, consulta i seguenti argomenti:<br>• Assegnazione di più indirizzi IP ai nodi Amazon EKS con prefissi<br>• Implementazione dei pod in sottoreti alternative con reti personalizzate<br>• Implementazione di cluster privati con accesso limitato a Internet<br>• Specificare un AMI |
| Dimensioni del disco in Configurazione di calcolo del gruppo di nodi nella pagina Imposta configurazione di calcolo e dimensionamento: la console mostra il messaggio Specificato nel modello di lancio. | Dimensioni in Archiviazione (volumi) (Aggiungi nuovo volume). È necessario specificarlo nel modello di avvio. |
| Coppia di chiavi SSH in Configurazione del gruppo di nodi nella pagina Specifica reti: la console mostra la chiave specificata nel modello di avvio o mostra il messaggio Non specificato nel modello di lancio. | Nome della coppia di chiavi in Coppia di chiavi (login). |
| Non è possibile specificare gruppi di sicurezza di origine a cui è consentito l'accesso remoto quando si utilizza un modello di avvio. | Gruppi di sicurezza in Impostazioni di rete per l'istanza o Gruppi di sicurezza in Interfacce di rete (Aggiungi interfaccia di rete), ma non entrambi. Per ulteriori informazioni, consulta the section called "Utilizzo di gruppi di sicurezza personalizzati". |

Note

- Se si esegue l'implementazione di un gruppo di nodi utilizzando un modello di avvio, specificare uno o nessun Tipo di istanza in Contenuti del modello di avvio in un modello di avvio. In alternativa, è possibile specificare 0-20 tipi di istanza per Tipi di istanza nella pagina Impostare la configurazione di calcolo e dimensionamento della console. In alternativa, è possibile farlo utilizzando altri strumenti che utilizzano l'API Amazon EKS. Se specifichi un tipo di istanza in un modello di avvio e si utilizza tale modello di avvio per implementare il gruppo di nodi, non è possibile specificare alcun tipo di istanza nella console o utilizzare altri strumenti che sfruttino l'API Amazon EKS. Se non specifichi un tipo di istanza in un modello di avvio nella console o in altri strumenti che utilizzano l'API Amazon EKS, il tipo di istanza `t3.medium` sarà utilizzato. Se il gruppo di nodi utilizza il tipo di capacità Spot, è consigliabile specificare più tipi di istanza utilizzando la console. Per ulteriori informazioni, consulta the section called "Tipi di capacità del gruppo di nodi gestiti".
- Se i container che si implementano nel gruppo di nodi utilizzano Instance Metadata Service versione 2, assicurarsi di impostare il Limite hop di risposta metadati a 2 nel modello di avvio. Per ulteriori informazioni, consulta Metadati e dati dell'utente delle istanze nella Guida per l'utente di Amazon EC2.
- I modelli di avvio non supportano la funzionalità `InstanceRequirements` che consente la selezione flessibile del tipo di istanza.

## Assegnazione di tag a istanze Amazon EC2

É possibile utilizzare il parametro `TagSpecification` di un modello di avvio per specificare quali tag applicare alle istanze Amazon EC2 nel gruppo di nodi. L'entità IAM che chiama le API `CreateNodegroup` o `UpdateNodegroupVersion` devono disporre delle autorizzazioni per `ec2:RunInstances` e `ec2:CreateTags`, e i tag devono essere aggiunti al modello di avvio.

## Utilizzo di gruppi di sicurezza personalizzati

É possibile utilizzare un modello di avvio per specificare i gruppi di sicurezza Amazon EC2 da applicare alle istanze del gruppo di nodi. Questo può essere nel parametro gruppi di sicurezza a livello di istanza o come parte dei parametri di configurazione dell'interfaccia di rete. Non è possibile avviare un'istanza da un modello di avvio che specifica sia il livello di istanza sia l'interfaccia di rete dei gruppi di sicurezza. Considerare le seguenti condizioni che si applicano all'utilizzo di gruppi di sicurezza personalizzati con gruppi di nodi gestiti:

- Quando si utilizza Console di gestione AWS, Amazon EKS consente solo modelli di avvio con una singola specifica di interfaccia di rete.
- Per impostazione predefinita, Amazon EKS applica il Gruppo di sicurezza del cluster alle istanze nel gruppo di nodi, per facilitare la comunicazione tra i nodi e il piano di controllo. Se specifichi gruppi di sicurezza personalizzati nel modello di avvio utilizzando entrambe le opzioni menzionate in precedenza, Amazon EKS non aggiunge il gruppo di sicurezza del cluster. Pertanto, è necessario assicurarsi che le regole in entrata e in uscita dei gruppi di sicurezza abilitino la comunicazione con l'endpoint del cluster. Se le regole del gruppo di sicurezza non sono corrette, i nodi (worker) non possono aggiungersi al cluster. Per ulteriori informazioni sulle regole del gruppo di sicurezza, consultare the section called "Requisiti relativi al gruppo di sicurezza".
- Se è necessario l'accesso SSH alle istanze nel gruppo di nodi, includere un gruppo di sicurezza che consenta tale accesso.

## Dati utente Amazon EC2

Il modello di avvio include una sezione per i dati utente personalizzati. È possibile specificare le impostazioni di configurazione per il gruppo di nodi in questa sezione senza creare manualmente singole AMI personalizzate. Per ulteriori informazioni sulle impostazioni disponibili per Bottlerocket, consulta Using user data on. GitHub

É possibile fornire i dati utente Amazon EC2 nel tuo modello di avvio utilizzando `cloud-init` durante l'avvio delle istanze. Per ulteriori informazioni, consultare la documentazione cloud-init. I dati utente possono essere utilizzati per eseguire operazioni di configurazione comuni. Sono comprese le seguenti opzioni:

- Inclusione di utenti o gruppi.
- Installazione di pacchetti

I dati utente di Amazon EC2 nei modelli di avvio utilizzati con i gruppi di nodi gestiti devono essere nel formato Archivio MIME in più parti per le AMI Amazon Linux e nel formato TOML per le AMI Bottlerocket. Questo perché i dati utente vengono uniti con i dati utente Amazon EKS necessari ai nodi per aderire al cluster. Non specificare alcun comando nei dati utente che avviano o modificano `kubelet`. Questa operazione viene eseguita come parte dei dati utente uniti da Amazon EKS. Certi parametri `kubelet`, come l'impostazione delle etichette sui nodi, possono essere configurati direttamente tramite l'API dei gruppi di nodi gestiti.

> **Note**
>
> Per ulteriori informazioni sulle personalizzazioni `kubelet` avanzate, tra cui l'avvio manuale o il passaggio a parametri di configurazione personalizzati, vedere the section called "Specifica di un'AMI". Amazon EKS non unisce i dati utente se un ID AMI viene specificato in un modello di avvio.

I seguenti dettagli forniscono ulteriori informazioni sulla sezione dei dati utente.

Dati utente di Amazon Linux 2

È possibile unire più blocchi di dati utente in un unico blocco, detto file MIME in più parti. Ad esempio, è possibile combinare un hook di avvio del cloud per la configurazione del daemon Docker con uno script di shell dei dati utente che installa un pacchetto personalizzato. Un file MIME in più parti è composto dai seguenti elementi:

- Il tipo di contenuto e la dichiarazione di delimitazione della parte: `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- La dichiarazione della versione MIME: `MIME-Version: 1.0`
- Uno o più blocchi di dati utente, che contengono i seguenti elementi:
  - La delimitazione di apertura, che indica l'inizio di un blocco di dati utente: `--==MYBOUNDARY==`
  - La dichiarazione del tipo di contenuto per il blocco: `Content-Type: text/cloud-config; charset="us-ascii"`. Per ulteriori informazioni sui tipi di contenuto, consultare la documentazione di cloud-init.
  - Il contenuto dei dati utente, ad esempio un elenco di comandi shell o direttive `cloud-init`.
  - La delimitazione di chiusura, che indica la fine del file MIME in più parti: `--==MYBOUNDARY==--`

Di seguito è riportato un esempio di un file in più parti MIME che è possibile utilizzare per crearne uno.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
```

```
#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Dati utente di Amazon Linux 2023

Amazon Linux 2023 (AL2023) introduce un nuovo processo di inizializzazione dei nodi `nodeadm` che utilizza uno schema di configurazione YAML. Se utilizzi gruppi di nodi autogestiti o un'AMI con un modello di avvio, ora dovrai fornire esplicitamente metadati del cluster aggiuntivi quando crei un nuovo gruppo di nodi. Un esempio dei parametri minimi richiesti è come segue, dove ora `apiServerEndpoint`, `certificateAuthority` e il servizio `cidr` sono richiesti:

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

In genere questa configurazione è impostata nei dati utente, così com'è o incorporata in un documento MIME composto da più parti:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

In AL2, i metadati di questi parametri sono stati rilevati dalla chiamata `DescribeCluster` API di Amazon EKS. Con AL2023, questo comportamento è cambiato poiché la chiamata API aggiuntiva

rischia una limitazione (della larghezza di banda della rete) durante gli aumenti verticali su larga scala dei nodi. Questa modifica non ha effetto su di te se utilizzi gruppi di nodi gestiti senza un modello di avvio o se utilizzi Karpenter. Per ulteriori informazioni su `certificateAuthority` e sul servizio `cidr`, consulta `DescribeCluster` nel Riferimento di API Amazon EKS.

Ecco un esempio completo di dati utente AL2023 che combina uno script di shell per personalizzare il nodo (come l'installazione di pacchetti o la pre-memorizzazione nella cache delle immagini dei container) con la configurazione `nodeadm` richiesta. Questo esempio mostra personalizzazioni comuni, tra cui: * Installazione di pacchetti di sistema aggiuntivi * immagini dei Pre-caching contenitori per migliorare il tempo di avvio del Pod * Impostazione della configurazione del proxy HTTP * Configurazione `kubelet` dei flag per l'etichettatura dei nodi

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -o errexit
set -o pipefail
set -o nounset

# Install additional packages
yum install -y htop jq iptables-services

# Pre-cache commonly used container images
nohup docker pull public.ecr.aws/eks-distro/kubernetes/pause:3.2 &

# Configure HTTP proxy if needed
cat > /etc/profile.d/http-proxy.sh << 'EOF'
export HTTP_PROXY="http://proxy.example.com:3128"
export HTTPS_PROXY="http://proxy.example.com:3128"
export NO_PROXY="localhost,127.0.0.1,169.254.169.254,.internal"
EOF

--BOUNDARY
Content-Type: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
```

```
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      clusterDNS:
      - 10.100.0.10
    flags:
    - --node-labels=app=my-app,environment=production

--BOUNDARY--
```

Dati utente Bottlerocket

I dati utente delle strutture Bottlerocket nel formato TOML. È possibile fornire i dati utente da unire con i dati utente forniti da Amazon EKS. Ad esempio, è possibile fornire ulteriori impostazioni `kubelet`.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

Per ulteriori informazioni sulle impostazioni supportate, consultare la documentazione Bottlerocket. È possibile configurare etichette di nodi e taint nei dati utente. Consigliamo, tuttavia, di configurarle all'interno del gruppo di nodi. In questo caso, Amazon EKS applica queste configurazioni.

Quando i dati utente sono uniti, la formattazione non è mantenuta, ma il contenuto rimane invariato. La configurazione fornita nei dati utente sostituisce tutte le impostazioni configurate da Amazon EKS. Quindi, se imposti `settings.kubernetes.max-pods` o `settings.kubernetes.cluster-dns-ip`, questi valori nei dati utente sono applicati ai nodi.

Amazon EKS non supporta tutti i TOML validi. Di seguito è riportato un elenco di formati noti non supportati:

- Quote all'interno delle chiavi stimate: `'quoted "value"' = "value"`
- Quote evase nei valori: `str = "I'm a string. \"You can quote me\""`
- Galleggianti e numeri interi misti: `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`

- Tipi misti nelle matrici: `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- Intestazioni tra parentesi con chiavi stimate: `[foo."bar.baz"]`

Dati utente Windows

I dati utente di Windows utilizzano i comandi. PowerShell Quando crei un gruppo di nodi gestiti, i dati utente personalizzati si combinano con i dati utente gestiti da Amazon EKS. PowerShell I tuoi comandi vengono prima, seguiti dai comandi gestiti per i dati utente, il tutto all'interno di un unico `<powershell></powershell>` tag.

> **Important**
>
> Durante la creazione di gruppi di nodi Windows, Amazon EKS aggiorna il `aws-auth ConfigMap` file per consentire ai Linux-based nodi di unirsi al cluster. Il servizio non configura automaticamente le autorizzazioni per le AMI Windows. Se utilizzi i nodi Windows, dovrai gestire l'accesso tramite l'API della voce di accesso o aggiornando direttamente `aws-auth ConfigMap`. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

> **Note**
>
> Quando non è specificato alcun ID AMI nel modello di avvio, nei dati utente non utilizzare lo script bootstrap di Amazon EKS per Windows per configurare Amazon EKS.

Di seguito sono riportati dati utente di esempio.

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

## Specifica di un'AMI

Se si dispone di uno dei seguenti requisiti, specificare un ID AMI nel campo `ImageId` del modello di avvio. Selezionare il requisito di cui si dispone per ulteriori informazioni.

Example Fornisci i dati utente per passare argomenti al file **`bootstrap.sh`** incluso in un' Linux/Bottlerocket AMI ottimizzata per Amazon EKS

Il termine "bootstrapping" indica l'aggiunta di comandi che possono essere eseguiti all'avvio di un'istanza. Ad esempio, il bootstrap consente di utilizzare argomenti kubelet aggiuntivi. È possibile passare gli argomenti allo script `bootstrap.sh` utilizzando `eksctl` senza specificare un modello di avvio. Oppure è possibile farlo specificando le informazioni nella sezione dati utente di un modello di avvio.

eksctl senza specificare un modello di avvio

Crea un file denominato *`my-nodegroup.yaml`* con i seguenti contenuti. Sostituisci ogni *`example value`* con i valori in tuo possesso. Gli argomenti `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` sono facoltativi, ma grazie alla loro definizione lo script `bootstrap.sh` può evitare di effettuare una chiamata a `describeCluster`. Ciò è utile nelle configurazioni di cluster privati o nei cluster in cui si effettuano dimensionamenti frequenti dei nodi. Per ulteriori informazioni sullo `bootstrap.sh` script, consulta il file bootstrap.sh su GitHub.

- L'unico argomento richiesto è il nome del cluster (*`my-cluster`*).
- Per recuperare l'ID di un'AMI ottimizzata per `ami-`*`1234567890abcdef0`* , consulta le sezioni seguenti:
  - Recupera gli ID di AMI Amazon Linux consigliati
  - Recupera gli ID di AMI Bottlerocket consigliati
  - Recupera gli ID di AMI Microsoft Windows consigliati
- Per recuperare il valore *`certificate-authority`* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Per recuperare il valore *`api-server-endpoint`* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- Il valore per `--dns-cluster-ip` è il tuo servizio CIDR con `.10` alla fine. Per recuperare il valore *`service-cidr`* per il cluster, esegui il comando seguente. Ad esempio, se il valore restituito è `ipv4 10.100.0.0/16`, il tuo valore è *`10.100.0.10`*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Questo esempio fornisce un argomento `kubelet` per impostare un valore `max-pods` personalizzato utilizzando lo script `bootstrap.sh` incluso nell'AMI ottimizzata per Amazon EKS. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti. Per assistenza nella scelta di *`my-max-pods-value`*, consulta the section called "Come viene determinato MaxPods". Per ulteriori informazioni su come `maxPods` viene determinato quando si utilizzano i gruppi di nodi gestiti, vedere. the section called "Come viene determinato MaxPods"

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

Per ogni opzione disponibile per il file `eksctl config`, consulta Schema del file config nella documentazione su `eksctl`. L'utility `eksctl` crea autonomamente un modello di avvio e popola i dati utente con i dati forniti nel file `config`.

Crea un gruppo di nodi con il comando seguente.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

Dati utente in un modello di avvio

Specifica le seguenti informazioni nella sezione dati utente del modello di avvio. Sostituisci ogni *example value* con i valori in tuo possesso. Gli argomenti `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` sono facoltativi, ma grazie alla loro definizione lo script `bootstrap.sh` può evitare di effettuare una chiamata a `describeCluster`. Ciò è utile nelle configurazioni di cluster privati o nei cluster in cui si effettuano dimensionamenti frequenti dei nodi. Per ulteriori informazioni sullo `bootstrap.sh` script, vedere il file bootstrap.sh su GitHub.

- L'unico argomento richiesto è il nome del cluster (*my-cluster*).
- Per recuperare il valore *certificate-authority* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Per recuperare il valore *api-server-endpoint* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- Il valore per `--dns-cluster-ip` è il tuo servizio CIDR con `.10` alla fine. Per recuperare il valore *service-cidr* per il cluster, esegui il comando seguente. Ad esempio, se il valore restituito è `ipv4 10.100.0.0/16`, il tuo valore è *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Questo esempio fornisce un argomento `kubelet` per impostare un valore `max-pods` personalizzato utilizzando lo script `bootstrap.sh` incluso nell'AMI ottimizzata per Amazon EKS. Per assistenza nella scelta di *my-max-pods-value*, consulta the section called "Come viene determinato MaxPods". Per ulteriori informazioni su come `maxPods` viene determinato quando si utilizzano i gruppi di nodi gestiti, vederethe section called "Come viene determinato MaxPods".

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
```

```
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

Example Fornire i dati dell'utente per passare gli argomenti al file **`Start-EKSBootstrap.ps1`** incluso nell'AMI Windows ottimizzata per Amazon EKS

Il termine "bootstrapping" indica l'aggiunta di comandi che possono essere eseguiti all'avvio di un'istanza. È possibile passare gli argomenti allo script `Start-EKSBootstrap.ps1` utilizzando `eksctl` senza specificare un modello di avvio. Oppure è possibile farlo specificando le informazioni nella sezione dati utente di un modello di avvio.

Per specificare un ID AMI Windows personalizzato, fai le seguenti considerazioni:

- Devi utilizzare un modello di avvio e fornire i comandi bootstrap richiesti nella sezione dei dati utente. Per recuperare l'ID Windows desiderato, è possibile utilizzare la tabella in Crea nodi con AMI Windows ottimizzate.
- Esistono vari limiti e condizioni. Ad esempio, devi aggiungere `eks:kube-proxy-windows` alla tua mappa di configurazione AWS IAM Authenticator. Per ulteriori informazioni, consulta the section called "Limiti e condizioni quando si specifica un ID AMI".

Specifica le seguenti informazioni nella sezione dati utente del modello di avvio. Sostituisci ogni *`example value`* con i valori in tuo possesso. Gli argomenti `-APIServerEndpoint`, `-Base64ClusterCA` e `-DNSClusterIP` sono facoltativi, ma grazie alla loro definizione lo script `Start-EKSBootstrap.ps1` può evitare di effettuare una chiamata a `describeCluster`.

- L'unico argomento richiesto è il nome del cluster (*`my-cluster`*).
- Per recuperare il valore *`certificate-authority`* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- Per recuperare il valore *api-server-endpoint* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- Il valore per `--dns-cluster-ip` è il tuo servizio CIDR con `.10` alla fine. Per recuperare il valore *service-cidr* per il cluster, esegui il comando seguente. Ad esempio, se il valore restituito è `ipv4 10.100.0.0/16`, il tuo valore è *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- Per ulteriori argomenti, consulta.the section called "Parametri di configurazione dello script di bootstrap".

> **Note**
>
> Se utilizzi il servizio personalizzato CIDR, devi specificarlo utilizzando il parametro `-ServiceCIDR`. In caso contrario, la risoluzione DNS per i pod nel cluster avrà esito negativo.

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

Example Eseguire un'AMI personalizzata a causa di specifici requisiti di sicurezza, conformità o policy interne

Per ulteriori informazioni, consulta Amazon Machine Image (AMI) nella Guida per l'utente di Amazon EC2. La specifica di compilazione AMI Amazon EKS contiene risorse e script di configurazione per la creazione di un'AMI Amazon EKS personalizzata basata su Amazon Linux. Per ulteriori informazioni,

consulta Specifica della build AMI di Amazon EKS su GitHub. Per creare AMI personalizzate installate con altri sistemi operativi, consulta AMI personalizzate di esempio di Amazon EKS su GitHub.

Non è possibile utilizzare riferimenti dinamici ai parametri per gli ID AMI nei modelli di avvio utilizzati con gruppi di nodi gestiti.

⚠ Important

- Quando si specifica un'AMI, Amazon EKS non convalida la versione di Kubernetes incorporata nell'AMI rispetto alla versione del piano di controllo del cluster. Sei responsabile di garantire che la versione Kubernetes della tua AMI personalizzata sia conforme alla politica di modifica della versione di Kubernetes:
  - La `kubelet` versione sui tuoi nodi non deve essere più recente della versione del cluster
  - La `kubelet` versione sui tuoi nodi deve essere uguale o superiore a 3 versioni secondarie rispetto alla versione del cluster (per la versione Kubernetes `1.28` o successiva) o fino a 2 versioni minori rispetto alla versione del cluster (per la versione Kubernetes o precedente) `1.27`

    La creazione di gruppi di nodi gestiti con violazioni dell'asimmetria delle versioni può comportare:
  - I nodi non riescono a unirsi al cluster
  - Comportamento indefinito o incompatibilità delle API
  - Instabilità del cluster o errori del carico di lavoro
- Quando si specifica una AMI, Amazon EKS non unisce alcun dato utente. È l'utente, piuttosto, ad essere responsabile della fornitura dei comandi `bootstrap` richiesti per unire i nodi al cluster. Se i nodi non riescono a unirsi al cluster, le operazioni `CreateNodegroup` e `UpdateNodegroupVersion` di Amazon EKS non vengono eseguite con successo.

## Limiti e condizioni quando si specifica un ID AMI

Di seguito sono riportati i limiti e le condizioni per la specifica di un ID AMI con gruppi di nodi gestiti:

- È necessario creare un nuovo gruppo di nodi per passare dalla specifica o meno di un ID AMI in un modello di avvio.

- Non riceverai una notifica nella console quando è disponibile una versione AMI più recente. Per aggiornare il gruppo di nodi a una versione AMI più recente, devi creare una nuova versione del modello di avvio con un ID AMI aggiornato. Quindi, è necessario aggiornare il gruppo di nodi con la nuova versione del modello di avvio.
- I seguenti campi non possono essere impostati nell'API se si specifica un ID AMI:
  - `amiType`
  - `releaseVersion`
  - `version`
- Se si specifica un ID AMI, qualsiasi set di `taints` nell'API viene applicato in modo asincrono. Per applicare i taint prima che un nodo si unisca al cluster, è necessario passare i taint a `kubelet` utilizzando il flag della riga di comando `--register-with-taints`. Per ulteriori informazioni, consulta kubelet nella documentazione Kubernetes.
- Quando specifichi un ID AMI personalizzato per i gruppi di nodi gestiti da Windows, aggiungilo `eks:kube-proxy-windows` alla mappa di configurazione di AWS IAM Authenticator. Questa API è necessaria per il funzionamento di DNS.
  a. Apri la mappa di configurazione di AWS IAM Authenticator per modificarla.

     ```
     kubectl edit -n kube-system cm aws-auth
     ```

  b. Aggiungi questa voce all'elenco `groups` sotto ogni `rolearn` associato ai nodi Windows. La tua mappa di configurazione dovrebbe essere simile a aws-auth-cm-windows.yaml.

     ```
     - eks:kube-proxy-windows
     ```

  c. Salva il file ed esci dall'editor di testo.
- Per ogni AMI che utilizza un modello di avvio personalizzato, l'impostazione predefinita `HttpPutResponseHopLimit` per i gruppi di nodi gestiti è impostata su 2.

## Eliminare un gruppo di nodi gestito dal cluster

In questo argomento viene descritto come eliminare un gruppo di nodi gestiti Amazon EKS. Quando elimini un gruppo di nodi gestiti, Amazon EKS imposterà su zero la dimensione minima, massima e desiderata del tuo gruppo Auto Scaling. Questo fa sì che il gruppo di nodi venga dimensionato.

Prima che ogni istanza venga terminata, Amazon EKS invia un segnale per drenare quel nodo. Durante il processo di drenaggio, Kubernetes esegue le seguenti operazioni per ogni pod sul nodo:

esegue tutti gli hook del `preStop` ciclo di vita configurati, invia `SIGTERM` segnali ai contenitori, quindi attende il corretto spegnimento. `terminationGracePeriodSeconds` Se il nodo non è stato svuotato dopo 5 minuti, Amazon EKS consente ad Auto Scaling di continuare la chiusura forzata dell'istanza. Dopo che tutte le istanze sono state terminate, il gruppo Auto Scaling viene eliminato.

> **Important**
>
> Se si elimina un gruppo di nodi gestiti che utilizza un ruolo IAM del nodo non impiegato da nessun altro gruppo di nodi gestiti nel cluster, il ruolo viene rimosso dalla `ConfigMap` `aws-auth`. Se i gruppi di nodi autogestiti nel cluster utilizzano lo stesso ruolo IAM del nodo, i nodi autogestiti passeranno allo stato `NotReady`. Inoltre, anche l'operazione del cluster viene interrotta. Per aggiungere una mappatura per il ruolo da utilizzare solo per i gruppi di nodi autogestiti, consultare la pagina the section called "Creare voci di accesso", se la versione della piattaforma del cluster corrisponde almeno alla versione minima elencata nella sezione dei prerequisiti della pagina Grant IAM users access to Kubernetes with EKS access entries. Se la versione della piattaforma è precedente alla versione minima richiesta per le voci di accesso, è possibile aggiungere nuovamente la voce alla `ConfigMap` `aws-auth`. Per ulteriori informazioni, digita `eksctl create iamidentitymapping --help` nel terminale.

È possibile creare un gruppo di nodi gestiti con i seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

## `eksctl`

Eliminare un gruppo di nodi gestiti con **`eksctl`**

Inserire il seguente comando. Sostituisci ogni `<example value>` con i valori in tuo possesso.

```
eksctl delete nodegroup \
  --cluster <my-cluster> \
  --name <my-mng> \
  --region <region-code>
```

Per ulteriori opzioni, consulta Eliminazione e svuotamento dei gruppi di nodi nella documentazione di `eksctl`.

### Console di gestione AWS

Eliminare un gruppo di nodi gestiti con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nella pagina Cluster, scegli il cluster che contiene il gruppo di nodi da eliminare.
3. Nella pagina del cluster selezionata, scegli la scheda Calcolo.
4. Nella sezione Gruppi di nodi, scegliere il gruppo di nodi da eliminare. Scegli Elimina.
5. Nella finestra di dialogo di conferma di Eliminazione del gruppo di nodi, inserisci il nome del gruppo di nodi. Scegli Elimina.

### AWS CLI

Eliminare un gruppo di nodi gestito con AWS CLI

1. Inserire il seguente comando. Sostituisci ogni `<example value>` con i valori in tuo possesso.

```
aws eks delete-nodegroup \
  --cluster-name <my-cluster> \
  --nodegroup-name <my-mng> \
  --region <region-code>
```

2. Se `cli_pager=` è impostato nella configurazione CLI, usa i tasti freccia sulla tastiera per scorrere l'output della risposta. Premere il tasto q al termine.

   Per ulteriori opzioni, consultate il `delete-nodegroup` comando nella AWS CLI Command Reference.

## Gestione autonoma dei nodi con nodi autogestiti

Un cluster contiene uno o più nodi Amazon EC2 su cui sono pianificati i pod. I nodi Amazon EKS vengono eseguiti nell'account AWS e si connettono al piano di controllo del cluster tramite l'endpoint del server API del cluster. La loro fatturazione è basata sui prezzi Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.

Un cluster può contenere diversi gruppi di nodi. Ciascun gruppo di nodi contiene uno o più nodi implementati in un gruppo Amazon EC2 Auto Scaling. Il tipo di istanza dei nodi all'interno del gruppo può variare, ad esempio quando si utilizza la selezione del tipo di istanza basata sugli attributi con Karpenter. Tutte le istanze di un gruppo di nodi devono utilizzare il ruolo IAM del nodo Amazon EKS.

Amazon EKS fornisce Amazon Machine Image (AMI) specializzate chiamate AMI ottimizzate per Amazon EKS. Le AMI sono configurate per funzionare con Amazon EKS. I loro componenti includono `containerd`, `kubelet` e l'AWS IAM Authenticator. L'AMI contiene anche uno script di bootstrap specializzato che consente di individuare e connettersi automaticamente al piano di controllo del cluster.

Se si limita l'accesso all'endpoint pubblico del cluster utilizzando blocchi CIDR, è consigliabile abilitare anche l'accesso agli endpoint privati. In questo modo i nodi possono comunicare con il cluster. Senza l'endpoint privato abilitato, i blocchi CIDR specificati per l'accesso pubblico devono includere le origini di uscita dal VPC. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

Per aggiungere nodi autogestiti al cluster Amazon EKS, consultare gli argomenti che seguono. Se si avviano manualmente i nodi autogestiti, è necessario aggiungere il seguente tag a ciascun nodo, assicurandosi al contempo che `<cluster-name>` corrisponda al cluster. Per ulteriori informazioni, consultare Aggiunta ed eliminazione di tag in una singola risorsa. Se si seguono i passaggi della guida, il tag richiesto viene aggiunto al nodo per conto dell'utente.

| Chiave | Valore |
| --- | --- |
| `kubernetes.io/cluster/<cluster-name>` | `owned` |

**Important**

I tag nel servizio di metadati di istanza (IMDS) di Amazon EC2 non sono compatibili con i nodi EKS. Quando i tag dei metadati delle istanze sono abilitati, è impedito l'uso di barre ("/") nei valori dei tag. Tale limitazione può causare errori nell'avvio delle istanze, in particolare quando si utilizzano strumenti di gestione dei nodi come Karpenter o un dispositivo di dimensionamento automatico dei cluster, poiché questi servizi si basano solo su tag contenenti barre per una corretta funzionalità.

Per ulteriori informazioni sui nodi da un prospettiva Kubernetes generale, consultare Nodi nella documentazione Kubernetes.

Argomenti


- Crea nodi Amazon Linux autogestiti
- Crea nodi autogestiti Bottlerocket
- Crea nodi Microsoft Windows autogestiti
- Creazione di nodi Ubuntu Linux autogestiti
- Aggiornamento dei nodi autogestiti per il tuo cluster


## Crea nodi Amazon Linux autogestiti

In questo argomento viene descritto come avviare gruppi con scalabilità automatica di nodi Linux che si registrano con il cluster Amazon EKS. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi. Puoi anche avviare nodi Amazon Linux autogestiti con `eksctl` o. Console di gestione AWS Se devi avviare nodi su AWS Outposts, consulta. the section called "Nodi"

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta the section called "Creazione di un cluster ". Se hai delle sottoreti nella AWS regione in cui hai abilitato AWS Outposts, AWS Wavelength o AWS Local Zones, tali sottoreti non devono essere state passate al momento della creazione del cluster.
- Un ruolo IAM esistente per i nodi da utilizzare. Per crearne uno, consulta the section called "Ruolo IAM del nodo". Se questo ruolo non prevede nessuna delle policy per VPC CNI, per i pod VPC CNI è necessario il ruolo separato riportato di seguito.
- (Facoltativo, ma consigliato) Il componente aggiuntivo plug-in CNI di Amazon VPC per Kubernetes configurato con il proprio ruolo IAM a cui è allegata la policy IAM necessaria. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
- Familiarità con le considerazioni presenti alla pagina Choose an optimal Amazon EC2 node instance type. A seconda del tipo di istanza scelto, potrebbero esserci ulteriori prerequisiti per il cluster e il VPC.

È possibile avviare nodi Linux autogestiti con uno dei seguenti modi:

- the section called "`eksctl`"

- the section called "Console di gestione AWS"

### `eksctl`

Avvia nodi Linux autogestiti utilizzando **`eksctl`**

1. Installa la versione `0.215.0` o una versione successiva dello strumento da riga di `eksctl` comando installato sul tuo dispositivo o. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. (Facoltativo) Se la policy IAM gestita AmazonEKS_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, ti consigliamo, invece, di assegnarla a un ruolo IAM associato all'account di servizio Kubernetes `aws-node`. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
3. Il comando seguente crea un gruppo di nodi in un cluster esistente. Sostituisci *`al-nodes`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti. Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e dell' AWS account in cui stai creando il cluster. Sostituisci i *`example value`* rimanenti con i valori in tuo possesso. Per impostazione predefinita, i nodi vengono creati con la stessa versione Kubernetes del piano di controllo.

   Prima di scegliere un valore per `--node-type`, consulta Choose an optimal Amazon EC2 node instance type.

   Sostituisci *`my-key`* con il nome della coppia di chiavi Amazon EC2 o della chiave pubblica. Questa chiave viene utilizzata per eseguire il SSH nei nodi dopo il loro avvio. Se non disponi di una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta la sezione relativa alle coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2.

   Crea il tuo gruppo di nodi con il comando seguente.

   > **Important**
   >
   > Se desideri distribuire un gruppo di nodi nelle sottoreti AWS Outposts, Wavelength o Local Zone, ci sono altre considerazioni:

- Le sottoreti non devono essere state trasmesse al momento della creazione del cluster.
- È necessario creare il gruppo di nodi con un file di configurazione, specificando le sottoreti e `volumeType: gp2`. Per ulteriori informazioni, consulta Creazione di un gruppo di nodi da un file di configurazione e lo Schema del file config nella documentazione di `eksctl`.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

Per implementare un gruppo di nodi che:

- può assegnare un numero significativamente più elevato di indirizzi IP a pod rispetto alla configurazione predefinita, consulta the section called "Aumentare gli indirizzi IP".
- può assegnare gli indirizzi `IPv4` a pod da un altro intervallo CIDR rispetto a quello dell'istanza, consulta the section called "Rete personalizzata".
- può assegnare gli indirizzi `IPv6` a pod e servizi, consulta the section called "IPv6".
- non dispone di accesso a Internet in uscita, consulta the section called "Cluster privati".

  Per un elenco completo di tutte le opzioni e i valori predefiniti disponibili, immetti il comando seguente.

  ```
  eksctl create nodegroup --help
  ```

  Se i nodi di lavoro non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" nel capitolo sulla risoluzione dei problemi.

  Di seguito viene riportato un output di esempio. Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.
5. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

   Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

## Console di gestione AWS

Passaggio 1: avvia i nodi Linux autogestiti utilizzando la Console di gestione AWS

1. Scarica l'ultima versione del modello. AWS CloudFormation

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-26/
amazon-eks-nodegroup.yaml
```

2. Attendi che lo stato del cluster risulti `ACTIVE`. Se vengono avviati prima che il cluster sia attivo, i nodi non riescono a effettuare la registrazione al cluster e sarà necessario riavviarli.
3. Apri la AWS CloudFormation console.
4. Scegli Crea stack e quindi seleziona Con nuove risorse (standard).
5. In Specifica modello, seleziona Carica un file di modello e Scegli file.
6. Seleziona il file `amazon-eks-nodegroup.yaml` scaricato.
7. Seleziona Successivo.
8. Nella pagina Specifica i dettagli dello stack, immetti i parametri seguenti e scegli Successivo:
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *`my-cluster-nodes`*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.

- ClusterName: inserisci il nome che hai usato per creare il cluster Amazon EKS. Questo nome deve corrispondere a quello del cluster o i nodi non verranno aggiunti al cluster.
- ClusterControlPlaneSecurityGroup: scegli il SecurityGroupsvalore dall' AWS CloudFormation output che hai generato quando hai creato il tuo VPC.

  Nella procedura seguente viene illustrata un'operazione per recuperare il gruppo applicabile.

  a. Aprire la Console Amazon EKS.

  b. Scegli il nome del cluster.

  c. Scegli la scheda Reti.

  d. Utilizza il valore dei gruppi di sicurezza aggiuntivi come riferimento quando selezioni dall'ClusterControlPlaneSecurityGroupelenco a discesa.
- ApiServerEndpoint: Inserite l'API Server Endpoint per il vostro cluster EKS. È possibile trovarlo nella sezione Dettagli della console EKS Cluster
- CertificateAuthorityData: Immettete i dati dell'Autorità di certificazione codificati in base 64, disponibili anche nella sezione Dettagli della console del cluster EKS.
- ServiceCidr: Inserisci l'intervallo CIDR utilizzato per allocare gli indirizzi IP ai servizi Kubernetes all'interno del cluster. Si trova nella scheda di rete della console del cluster EKS.
- AuthenticationMode: Seleziona la modalità di autenticazione in uso nel cluster EKS esaminando la scheda di accesso all'interno della console del cluster EKS.
- NodeGroupName: Inserisci un nome per il tuo gruppo di nodi. Questo nome può essere utilizzato in seguito per identificare il gruppo di nodi con dimensionamento automatico creato per i tuoi nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti.
- NodeAutoScalingGroupMinSize: Inserisci il numero minimo di nodi su cui il gruppo Auto Scaling del nodo può scalare.
- NodeAutoScalingGroupDesiredCapacity: Inserisci il numero di nodi desiderato su cui scalare quando viene creato lo stack.
- NodeAutoScalingGroupMaxSize: Inserisci il numero massimo di nodi su cui il gruppo Auto Scaling del nodo può scalare orizzontalmente.
- NodeInstanceType: scegli un tipo di istanza per i tuoi nodi. Per ulteriori informazioni, consulta the section called "Tipi di istanza Amazon EC2".
- NodeImageIdSSMParam: Pre-populated con il parametro Amazon EC2 Systems Manager di una recente AMI Amazon Linux 2023 ottimizzata per Amazon EKS per una versione variabile di

Kubernetes. Se si desidera utilizzare una diversa versione secondaria di Kubernetes supportata da Amazon EKS, è possibile sostituire *1.XX* con una versione supportata differente. Si consiglia di specificare la stessa versione Kubernetes del cluster.

Puoi anche sostituirlo *amazon-linux-2023* con un altro tipo di AMI. Per ulteriori informazioni, consulta the section called "Ottieni le ultime novità IDs".

> **Note**
>
> L'AMI del nodo Amazon EKS è basata su Amazon Linux. Puoi tenere traccia degli eventi di sicurezza o privacy per Amazon Linux 2023 nell'Amazon Linux Security Center o abbonarti al feed RSS associato. Gli eventi di sicurezza e privacy includono una panoramica del problema, quali sono i pacchetti interessati e come aggiornare le istanze per risolvere il problema.

- NodeImageId: (Facoltativo) Se utilizzi un'AMI personalizzata (anziché un'AMI ottimizzata per Amazon EKS), inserisci un ID AMI del nodo per la tua AWS regione. Se specifichi un valore qui, questo sostituisce tutti i valori nel NodeImageIdSSMParamcampo.
- NodeVolumeSize: Specificate la dimensione del volume root per i nodi, in GiB.
- NodeVolumeType: Specificate un tipo di volume root per i nodi.
- KeyName: inserisci il nome di una coppia di chiavi SSH Amazon EC2 che puoi usare per connetterti tramite SSH ai tuoi nodi dopo il loro avvio. Se non hai già una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta la sezione relativa alle coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2.
- VpcId: inserisci l'ID per il VPC che hai creato.
- Sottoreti: scegli le sottoreti create per il VPC. Se hai creato il VPC utilizzando i passaggi descritti in Create an Amazon VPC for your Amazon EKS cluster, specifica solo le sottoreti private all'interno del VPC per avviare i nodi. È possibile visualizzare le sottoreti private aprendo ogni collegamento relativo alla sottorete dalla scheda Reti del cluster.

> **Important**
>
> - Se una qualsiasi delle sottoreti è pubblica, devi abilitare l'impostazione di assegnazione automatica degli indirizzi IP pubblici. Se l'impostazione non è abilitata per la sottorete pubblica, a tutti i nodi distribuiti in quella sottorete pubblica non verrà assegnato un indirizzo IP pubblico e non saranno in grado di comunicare con il cluster o altri servizi. AWS Se la sottorete è stata distribuita prima del 26 marzo

2020 utilizzando uno dei modelli AWS CloudFormation VPC di Amazon EKS o utilizzando`eksctl`, l'assegnazione automatica degli indirizzi IP pubblici è disabilitata per le sottoreti pubbliche. Per informazioni su come abilitare l'assegnazione di indirizzi IP pubblici per una sottorete, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete. Se il nodo viene distribuito in una sottorete privata, è in grado di comunicare con il cluster e altri servizi tramite un gateway NAT. AWS

- Se le sottoreti non hanno accesso a Internet, leggi con attenzione le considerazioni e i passaggi aggiuntivi descritti in Deploy private clusters with limited internet access.
- Se si AWS selezionano le sottoreti Outposts, Wavelength o Local Zone, le sottoreti non devono essere state passate al momento della creazione del cluster.

9. Seleziona le opzioni desiderate nella pagina Configura opzioni dello stack, quindi scegli Next (Avanti).
10. Seleziona la casella di controllo a sinistra di Riconosco che potrebbe creare risorse IAM. AWS CloudFormation , quindi scegli Create stack.
11. Al termine della creazione dello stack, selezionalo nella console e scegli Output. Se utilizzi le `EKS API` o le modalità di `EKS API and ConfigMap` autenticazione, questo è l'ultimo passaggio.
12. Se stai usando la modalità di `ConfigMap` autenticazione, registra il NodeInstanceRoleper il gruppo di nodi che è stato creato.

Fase 2: abilitazione dell'aggiunta di nodi al cluster

> **Note**
>
> I due passaggi seguenti sono necessari solo se si utilizza la modalità di autenticazione Configmap all'interno del cluster EKS. Inoltre, se hai avviato nodi all'interno di un VPC privato senza accesso a Internet in uscita, assicurati di consentire ai nodi di unirsi al cluster dall'interno del VPC.

1. Verifica se disponi già di una `ConfigMap` per `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrata una `ConfigMap` per `aws-auth`, aggiornala se necessario.

a. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Aggiungi una nuova voce `mapRoles`, se necessario. Impostate il `rolearn` valore sul NodeInstanceRolevalore registrato nella procedura precedente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. Salva il file ed esci dall'editor di testo.

3. Se hai ricevuto un errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`, applica lo `ConfigMap` stack.

a. Scarica la mappa di configurazione.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. Nel `aws-auth-cm.yaml` file, impostate il `rolearn` valore sul NodeInstanceRolevalore registrato nella procedura precedente. Per eseguire questa operazione, utilizza un editor di testo o sostituisci *`my-node-instance-role`* eseguendo il comando seguente:

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

Inserisci `Ctrl+C` per tornare a un prompt dello shell (interprete di comandi).

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso non autorizzato o negato (kubectl) " nell'argomento relativo alla risoluzione dei problemi.

Se i nodi di lavoro non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" nel capitolo sulla risoluzione dei problemi.

5. (Solo nodi GPU) Se hai scelto un tipo di istanza GPU e l'AMI accelerata ottimizzata per Amazon EKS, devi applicare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo *vX.X.X* con la versione desiderata prima di eseguire il comando seguente. NVIDIA/k8s-device-plugin

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Fase 3: Azioni aggiuntive

1. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.
2. (Facoltativo) Se la policy IAM gestita da Amazoneks_CNI_Policy (se hai `IPv4` un cluster) o la (che hai creato tu stesso se hai un `IPv6` cluster) è collegata al ruolo IAM *AmazonEKS_CNI_IPv6_Policy* del nodo Amazon EKS, ti consigliamo invece di assegnarlo a un ruolo IAM da associare all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
3. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

   Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

# Crea nodi autogestiti Bottlerocket

> **Note**
>
> I gruppi di nodi gestiti potrebbero offrire alcuni vantaggi per il tuo caso d'uso. Per ulteriori informazioni, consulta the section called "Gruppi di nodi gestiti".

Questo argomento descrive l'avvio di gruppi Auto Scaling di nodi Bottlerocket che si registrano con il cluster Amazon EKS. Bottlerocket è un sistema operativo Linux-based open source AWS che puoi utilizzare per eseguire contenitori su macchine virtuali o host bare metal. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi. Per ulteriori informazioni su Bottlerocket, consulta Using a Bottlerocket AMI with Amazon EKS on e Custom AMI support nella documentazione. GitHub https://eksctl.io/usage/custom-ami-support/ `eksctl`

Per informazioni sugli aggiornamenti in loco, consulta Bottlerocket Update Operator su. https://github.com/bottlerocket-os/bottlerocket-update-operator GitHub

> **Important**
>
> - I nodi di lavoro Amazon EKS sono istanze Amazon EC2 standard e la loro fatturazione è basata sui normali prezzi dell'istanza Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.
> - Puoi avviare i nodi Bottlerocket nei cluster estesi di Amazon EKS su AWS Outposts, ma non puoi avviarli nei cluster locali su Outposts. AWS Per ulteriori informazioni, consulta Amazon EKS su AWS Outposts.
> - Puoi eseguire l'implementazione su istanze Amazon EC2 con processori `x86` o Arm. Tuttavia, non puoi eseguire l'implementazione su istanze con chip Inferentia.
> - Bottlerocket AWS CloudFormation è compatibile con. Tuttavia, non esiste un CloudFormation modello ufficiale che possa essere copiato per distribuire i nodi Bottlerocket per Amazon EKS.
> - Le immagini Bottlerocket non vengono fornite con un server SSH o una shell. È possibile utilizzare metodi di accesso fuori banda per consentire a SSH di abilitare il container amministratore e passare alcuni passaggi di configurazione di bootstrap con i dati utente. Per ulteriori informazioni, consulta queste sezioni di Bottlerocket su: README.md GitHub
>   - Esplorazione

- Container amministratore
- Impostazioni Kubernetes

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

> **Note**
>
> Questa procedura funziona solo per i cluster creati con `eksctl`.

1. Copia i seguenti contenuti sul dispositivo. Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster. Sostituisci *`ng-bottlerocket`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e trattini bassi per i caratteri rimanenti. Per implementare su istanze Arm, sostituire *`m5.large`* con un tipo di istanza Arm. Sostituisci *`my-ec2-keypair-name`* con il nome di una coppia di chiavi SSH Amazon EC2; che puoi utilizzare per connetterti utilizzando SSH nei nodi dopo l'avvio. Se non disponi già di una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta la sezione relativa alle coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2. Sostituisci tutti i valori esemplificativi rimanenti con i valori in tuo possesso. Dopo aver effettuato le sostituzioni, esegui il comando modificato per creare il file `bottlerocket.yaml`.

   Se specifichi un tipo di istanza Amazon EC2 Arm, esamina le considerazioni in Amazon EKS optimized Arm Amazon Linux AMIs prima dell'implementazione. Per istruzioni su come eseguire la distribuzione utilizzando un'AMI personalizzata, consulta Building Bottlerocket on GitHub e Custom AMI support nella documentazione. https://eksctl.io/usage/custom-ami-support/ `eksctl`

Per implementare un gruppo di nodi gestito, implementare un'AMI personalizzata utilizzando un modello di avvio. Per ulteriori informazioni, consulta the section called "Modelli di avvio".

> **Important**
>
> Per distribuire un gruppo di nodi su sottoreti AWS Outposts, AWS Wavelength o AWS Local Zone, non trasmettere le sottoreti AWS Outposts, AWS Wavelength o Local Zone quando crei il cluster. AWS È necessario specificare le sottoreti nell'esempio seguente. Per ulteriori informazioni, vedere Creare un gruppo di nodi da un file di configurazione e lo Schema del file config nella documentazione su `eksctl`. *`region-code`*Sostituiscilo con la regione in cui si trova il cluster. AWS

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.36'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
```

```
EOF
```

2. Implementare i nodi mediante il comando seguente.

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

Di seguito viene riportato un output di esempio:

Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facoltativo) Creare un volume persistente Kubernetes su un nodo Bottlerocket usando il plug-in CSI di Amazon EBS. Il driver Amazon EBS di default si basa su strumenti di file system che non sono inclusi in Bottlerocket. Per ulteriori informazioni sulla creazione di una classe di archiviazione utilizzando il driver, consultare the section called "Amazon EBS".
4. (Facoltativo) Per impostazione predefinita `kube-proxy` imposta il parametro kernel `nf_conntrack_max` su un valore predefinito che può differire da quello che Bottlerocket imposta originariamente all'avvio. Per mantenere l'impostazione di default di Bottlerocket, modifica la configurazione `kube-proxy` con il comando seguente.

```
kubectl edit -n kube-system daemonset kube-proxy
```

Aggiungi `--conntrack-max-per-core` e `--conntrack-min` agli argomenti `kube-proxy` nell'esempio seguente. Un'impostazione di `0` non implica alcun cambiamento.

```
      containers:
      - command:
        - kube-proxy
        - --v=2
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. (Facoltativo) implementare un'applicazione di esempio per testare i nodi Bottlerocket.
6. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.

- Nessun pod nel cluster richiede l'accesso al servizio di metadati delle istanze Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

## Crea nodi Microsoft Windows autogestiti

Questo argomento illustra l'avvio di gruppi con scalabilità automatica di nodi Windows che si registrano con il cluster Amazon EKS. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi.

> **Important**
>
> - I nodi di lavoro Amazon EKS sono istanze Amazon EC2 standard e la loro fatturazione è basata sui normali prezzi dell'istanza Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.
> - Puoi avviare nodi Windows nei cluster estesi di Amazon EKS su AWS Outposts, ma non puoi avviarli in cluster AWS locali su Outposts. Per ulteriori informazioni, consulta Amazon EKS su AWS Outposts.

Abilitare il supporto Windows per il cluster. Si consiglia di rivedere le considerazioni importanti prima di avviare un gruppo di nodi Windows. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

Puoi avviare nodi Windows autogestiti con uno dei seguenti modi:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

### `eksctl`

Avvio dei nodi Windows autogestiti tramite **`eksctl`**

Questa procedura presuppone che `eksctl` sia installato e che la versione `eksctl` sia almeno `0.215.0`. È possibile verificare la tua versione con il seguente comando.

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

> **Note**
>
> Questa procedura funziona solo per i cluster creati con `eksctl`.

1. (Facoltativo) Se la policy IAM gestita da Amazoneks_CNI_Policy (se hai `IPv4` un cluster) o la (che hai creato tu stesso se hai un `IPv6` cluster) è collegata al ruolo IAM *`AmazonEKS_CNI_IPv6_Policy`* del nodo Amazon EKS, ti consigliamo invece di assegnarlo a un ruolo IAM da associare all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
2. In questa procedura si presuppone la disponibilità di un cluster. Se non disponi ancora di un cluster Amazon EKS e di un gruppo di nodi Amazon Linux a cui aggiungere un gruppo di nodi Windows, ti consigliamo di seguire the section called "Crea cluster (eksctl)". Questa guida fornisce una spiegazione dettagliata completa per la creazione di un cluster Amazon EKS con nodi Amazon Linux.

   Crea il tuo gruppo di nodi con il comando seguente. *`region-code`*Sostituiscilo con la regione in cui si trova il cluster. AWS Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster. Sostituisci *`ng-windows`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti. È possibile `2022` sostituire *`2019`* con Windows Server 2022 o utilizzare Windows Server `2025` 2025. Sostituisci il resto dei valori di esempio con quelli in tuo possesso.

   > **Important**
   >
   > Per distribuire un gruppo di nodi nelle sottoreti AWS Outposts, AWS Wavelength AWS o Local Zone, non passare le sottoreti AWS Outposts, Wavelength o Local Zone quando crei il cluster. Crea il gruppo di nodi con un file di configurazione, specificando

le sottoreti AWS Outposts, Wavelength o Local Zone. Per ulteriori informazioni, vedere Creazione di un gruppo di nodi da un file di configurazione e lo Schema del file config nella documentazione su `eksctl`.

```
eksctl create nodegroup \
    --region region-code \
    --cluster my-cluster \
    --name ng-windows \
    --node-type t2.large \
    --nodes 3 \
    --nodes-min 1 \
    --nodes-max 4 \
    --managed=false \
    --node-ami-family WindowsServer2019FullContainer
```

**Note**

- Se i nodi non riescono a unirsi al cluster, consultare the section called "Impossibile aggiungere i nodi al cluster" nella Guida alla risoluzione dei problemi.
- Per vedere le opzioni disponibili per i comandi `eksctl`, inserire il comando seguente.

  ```
  eksctl command -help
  ```

Di seguito viene riportato un output di esempio: Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opzionale) implementare un'applicazione di esempio per testare il cluster e i nodi Windows.
4. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

## Console di gestione AWS

### Prerequisiti

- Creazione di un cluster Amazon EKS e di un gruppo di nodi Linux. Se non disponi di tali risorse, ti consigliamo di crearle utilizzando una delle nostre guide in Nozioni di base. Queste guide descrivono come creare un cluster Amazon EKS con nodi Linux.
- Creazione di un VPC e di un gruppo di sicurezza esistenti che soddisfano i requisiti per un cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Requisiti del VPC e delle sottoreti" e the section called "Requisiti relativi al gruppo di sicurezza". Le guide in Nozioni di base consentono di creare un VPC che soddisfa i requisiti. In alternativa, è possibile anche seguire Create an Amazon VPC for your Amazon EKS cluster crearne uno nuovo manualmente.
- Un cluster Amazon EKS esistente che utilizza un VPC e un gruppo di sicurezza in grado di soddisfare i requisiti di un cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ". Se sono presenti sottoreti nella AWS regione in cui sono abilitati AWS Outposts, AWS Wavelength o AWS Local Zones, tali sottoreti non devono essere state passate al momento della creazione del cluster.

### Passaggio 1: avvia i nodi Windows autogestiti utilizzando la Console di gestione AWS

1. Attendi che lo stato del cluster risulti `ACTIVE`. Se i nodi vengono avviati prima che il cluster sia attivo, la registrazione al cluster non riesce e sarà necessario riavviarli.
2. Apri la console AWS CloudFormation
3. Seleziona Crea stack.
4. Per Specifica modello, selezionare URL Amazon S3.
5. Copia il seguente URL e incollalo nell'URL di Amazon S3.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Seleziona due volte Next (Avanti).

7. Nella pagina Quick create stack (Creazione rapida pila), inserire i seguenti parametri di conseguenza:
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo `my-cluster-nodes`.
   - ClusterName: inserisci il nome che hai usato per creare il cluster Amazon EKS.

     > **Important**
     >
     > Questo nome deve corrispondere esattamente a quello presente in Step 1: Create your Amazon EKS cluster. In caso contrario, i nodi non possono aggiungersi al cluster.

   - ClusterControlPlaneSecurityGroup: scegli il gruppo di sicurezza dall' AWS CloudFormation output che hai generato quando hai creato il tuo VPC. Nella procedura seguente viene illustrato un metodo per recuperare il gruppo applicabile.
     a. Aprire la Console Amazon EKS.
     b. Scegli il nome del cluster.
     c. Scegli la scheda Reti.
     d. Utilizza il valore dei gruppi di sicurezza aggiuntivi come riferimento quando selezioni dall'ClusterControlPlaneSecurityGroupelenco a discesa.
   - NodeGroupName: inserisci un nome per il tuo gruppo di nodi. Questo nome può essere utilizzato in seguito per identificare il gruppo di nodi con dimensionamento automatico creato per i tuoi nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti.
   - NodeAutoScalingGroupMinSize: Inserisci il numero minimo di nodi su cui il gruppo Auto Scaling del nodo può scalare.
   - NodeAutoScalingGroupDesiredCapacity: Inserisci il numero di nodi desiderato su cui scalare quando viene creato lo stack.
   - NodeAutoScalingGroupMaxSize: Inserisci il numero massimo di nodi su cui il gruppo Auto Scaling del nodo può scalare orizzontalmente.
   - NodeInstanceType: scegli un tipo di istanza per i tuoi nodi. Per ulteriori informazioni, consulta the section called "Tipi di istanza Amazon EC2".

> **Note**
>
> I tipi di istanza supportati per l'ultima versione del plug-in Amazon VPC CNI per Kubernetes sono elencati in vpc_ip_resource_limit.go on. GitHub Potrebbe essere necessario aggiornare la versione CNI per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

- NodeImageIdSSMParam: Pre-populated con il parametro Amazon EC2 Systems Manager dell'attuale ID AMI Windows Core ottimizzato per Amazon EKS. Per utilizzare la versione completa di Windows, sostituire `Core` con `Full`.
- NodeImageId: (Facoltativo) Se utilizzi un'AMI personalizzata (anziché un'AMI ottimizzata per Amazon EKS), inserisci un ID AMI del nodo per la tua AWS regione. Se specifichi un valore per questo campo, questo sostituisce tutti i valori presenti nel NodeImageIdSSMParamcampo.
- NodeVolumeSize: Specificate la dimensione del volume root per i nodi, in GiB.
- KeyName: inserisci il nome di una coppia di chiavi SSH Amazon EC2 che puoi usare per connetterti tramite SSH ai tuoi nodi dopo il loro avvio. Se non hai già una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta la sezione relativa alle coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2.

> **Note**
>
> Se non fornisci una key pair qui, lo AWS CloudFormation stack non viene creato.

- BootstrapArguments: Specificate eventuali argomenti opzionali da passare allo script di bootstrap del nodo, come ad esempio l'utilizzo di `kubelet` argomenti aggiuntivi. `-KubeletExtraArgs`
- DisableIMDSv1: ogni nodo supporta Instance Metadat Service versione 1 (IMDSv1) e IMDSv2 per impostazione predefinita. Puoi disabilitare IMDSv1. Per evitare che in futuro i nodi e i pod nel gruppo di nodi utilizzino IMDSv1, imposta DisableIMDSv1 su vero. Per ulteriori informazioni su IMDS, consulta Configurazione del servizio di metadati dell'istanza.
- VpcId: Seleziona l'ID per il VPC che hai creato.
- NodeSecurityGroups: Seleziona il gruppo di sicurezza che è stato creato per il tuo gruppo di nodi Linux quando hai creato il tuo VPC. Se ai nodi Linux è collegato più di un gruppo di sicurezza, specificarli tutti. Questo è il caso, ad esempio, se il gruppo di nodi Linux è stato creato con`eksctl`.

- Sottoreti: scegliere le sottoreti create. Se hai creato il VPC utilizzando i passaggi descritti in Create an Amazon VPC for your Amazon EKS cluster, specifica solo le sottoreti private all'interno del VPC per avviare i nodi.

> **Important**
>
> - Se una qualsiasi delle sottoreti è pubblica, devi abilitare l'impostazione di assegnazione automatica degli indirizzi IP pubblici. Se l'impostazione non è abilitata per la sottorete pubblica, a tutti i nodi distribuiti in quella sottorete pubblica non verrà assegnato un indirizzo IP pubblico e non saranno in grado di comunicare con il cluster o altri servizi. AWS Se la sottorete è stata distribuita prima del 26 marzo 2020 utilizzando uno dei modelli AWS CloudFormation VPC di Amazon EKS o utilizzando`eksctl`, l'assegnazione automatica degli indirizzi IP pubblici è disabilitata per le sottoreti pubbliche. Per informazioni su come abilitare l'assegnazione di indirizzi IP pubblici per una sottorete, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete. Se il nodo viene distribuito in una sottorete privata, è in grado di comunicare con il cluster e altri servizi tramite un gateway NAT. AWS
> - Se le sottoreti non hanno accesso a Internet, leggi con attenzione le considerazioni e i passaggi aggiuntivi descritti in Deploy private clusters with limited internet access.
> - Se si AWS selezionano le sottoreti Outposts, Wavelength o Local Zone, le sottoreti non devono essere state passate al momento della creazione del cluster.

8. Confermare che la pila sia in grado di creare risorse IAM, quindi scegliere Crea pila.
9. Al termine della creazione della pila, selezionala nella console e scegli Outputs (Uscite).
10. Registra il per il gruppo di nodi che è stato NodeInstanceRolecreato. Questo servirà durante la configurazione dei nodi Windows Amazon EKS.

Passaggio 2: abilita l'aggiunta di nodi al cluster

1. Verifica se disponi già di una `ConfigMap` per `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrata una `ConfigMap` per `aws-auth`, aggiornala se necessario.

   a. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Aggiungi nuove voci `mapRoles`, se necessario. Imposta i `rolearn` valori NodeInstanceRolesui valori che hai registrato nelle procedure precedenti.

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

c. Salva il file ed esci dall'editor di testo.

3. Se hai ricevuto un messaggio di errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`", applica lo `ConfigMap` di stock.

a. Scarica la mappa di configurazione.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm-windows.yaml
```

b. Nel `aws-auth-cm-windows.yaml` file, impostate i `rolearn` valori NodeInstanceRolesui valori applicabili registrati nelle procedure precedenti. Per eseguire questa operazione, utilizza un editor di testo o sostituisci i valori di esempio tramite il comando seguente:

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-node-
linux-instance-role|' \
    -e 's|<ARN of windows instance role (not instance profile)>|my-node-windows-
instance-role|' aws-auth-cm-windows.yaml
```

> **Important**
>
> - Non modificare altre righe in questo file.
> - Non utilizzare lo stesso ruolo IAM sia per i nodi Windows che per quelli Linux.

c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

Inserisci `Ctrl+C` per tornare a un prompt dello shell (interprete di comandi).

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso non autorizzato o negato (kubectl) " nell'argomento relativo alla risoluzione dei problemi.

Se i nodi di lavoro non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" nel capitolo sulla risoluzione dei problemi.

Fase 3: Azioni aggiuntive

1. (Opzionale) implementare un'applicazione di esempio per testare il cluster e i nodi Windows.
2. (Facoltativo) Se la policy IAM gestita da Amazoneks_CNI_Policy (se hai `IPv4` un cluster) o la (che hai creato tu stesso se hai un `IPv6` cluster) è collegata al ruolo IAM *`AmazonEKS_CNI_IPv6_Policy`* del nodo Amazon EKS, ti consigliamo invece di assegnarlo a un ruolo IAM da associare all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
3. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.

- Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

## Creazione di nodi Ubuntu Linux autogestiti

> **Note**
>
> I gruppi di nodi gestiti potrebbero offrire alcuni vantaggi per il tuo caso d'uso. Per ulteriori informazioni, consulta the section called "Gruppi di nodi gestiti".

Questo argomento illustra l'avvio di gruppi Auto Scaling di Ubuntu su nodi Amazon Elastic Kubernetes Service (EKS) o Ubuntu Pro su nodi Amazon Elastic Kubernetes Service (EKS) che si registrano con il cluster Amazon EKS. Ubuntu e Ubuntu Pro for EKS sono basati sull'Ubuntu Minimal LTS ufficiale, includono il AWS kernel personalizzato sviluppato congiuntamente e sono stati creati appositamente per EKS. AWS Ubuntu Pro aggiunge una copertura di sicurezza aggiuntiva supportando i periodi di supporto estesi di EKS, kernel livepatch, la conformità FIPS e la capacità di eseguire un numero illimitato di container Pro.

Dopo che i nodi sono aggiunti al cluster, è possibile implementare le applicazioni containerizzate per gli stessi. Per ulteriori informazioni, consulta la documentazione per Ubuntu su AWS e il supporto per le AMI personalizzate nella documentazione `eksctl`.

> **Important**
>
> - I nodi di lavoro Amazon EKS sono istanze Amazon EC2 standard e la loro fatturazione è basata sui normali prezzi dell'istanza Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.
> - Puoi avviare nodi Ubuntu nei cluster estesi di Amazon EKS su AWS Outposts, ma non puoi avviarli nei cluster locali su Outposts. AWS Per ulteriori informazioni, consulta Amazon EKS su AWS Outposts.

- È possibile eseguire l'implementazione su istanze Amazon EC2 con processori x86 o Arm. Tuttavia, le istanze che dispongono di chip Inferentia potrebbero dover installare prima l'SDK Neuron.

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

> **Note**
>
> Questa procedura funziona solo per i cluster creati con `eksctl`.

1. Copia i seguenti contenuti sul dispositivo. Sostituisci `my-cluster` con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfabetico e non può avere una lunghezza superiore a 100 caratteri. Sostituisci `ng-ubuntu` con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e trattini bassi per i caratteri rimanenti. Per implementare su istanze Arm, sostituire `m5.large` con un tipo di istanza Arm. Sostituisci `my-ec2-keypair-name` con il nome di una coppia di chiavi SSH Amazon EC2; che puoi utilizzare per connetterti utilizzando SSH nei nodi dopo l'avvio. Se non disponi già di una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2. Sostituisci tutti i valori esemplificativi rimanenti con i valori in tuo possesso. Dopo aver effettuato le sostituzioni, esegui il comando modificato per creare il file `ubuntu.yaml`.

   > **Important**
   >
   > Per distribuire un gruppo di nodi su sottoreti AWS Outposts, AWS Wavelength o AWS Local Zone, non trasmettere le sottoreti AWS Outposts, AWS Wavelength o Local Zone quando crei il cluster. AWS È necessario specificare le sottoreti nell'esempio seguente. Per ulteriori informazioni, vedere Creare un gruppo di nodi da un file di configurazione e lo

Schema del file config nella documentazione su `eksctl`. *`region-code`*Sostituiscilo con la regione in cui si trova il cluster. AWS

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.36'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Ubuntu2204
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

Per creare un gruppo di nodi Ubuntu Pro, basta cambiare il valore `amiFamily` in `UbuntuPro2204`.

2. Implementare i nodi mediante il comando seguente.

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

Di seguito viene riportato un output di esempio:

Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facoltativo) Implementa un'applicazione esemplificativa per testare i nodi Ubuntu.
4. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati delle istanze Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

   Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

## Aggiornamento dei nodi autogestiti per il tuo cluster

Quando viene rilasciata una nuova AMI ottimizzata per Amazon EKS, considerare la sostituzione dei nodi nel gruppo di nodi autogestiti con la nuova AMI. Analogamente, se hai aggiornato la versione Kubernetes per il tuo cluster Amazon EKS, aggiornare i nodi per l'utilizzo di nodi con la stessa versione Kubernetes.

> **Important**
>
> In questo argomento vengono descritti gli aggiornamenti dei nodi per i gruppi di nodi autogestiti. Se utilizzi gruppi di nodi gestiti, consulta the section called "Aggiorna".

Esistono due modi di base per aggiornare i gruppi di nodi autogestiti nei cluster per utilizzare una nuova AMI:

Migrate applications to a new node group

: crea un nuovo gruppo di nodi ed esegui la migrazione dei pod verso questo gruppo. La migrazione a un nuovo gruppo di nodi è più aggraziata del semplice aggiornamento dell'ID AMI in una pila AWS CloudFormation esistente. Questo perché il processo di migrazione esclude il

vecchio gruppo di nodi come `NoSchedule` e svuota i nodi dopo che una nuova pila è pronta ad accettare il carico di lavoro del pod esistente.

Aggiornamento di una pila di nodi AWS CloudFormation

Aggiorna la pila AWS CloudFormation per un gruppo di nodi esistente in modo che utilizzi la nuova AMI. Questo metodo non è supportato per i gruppi di nodi creati con `eksctl`.

## Esegui la migrazione delle applicazioni a un nuovo gruppo di nodi

Questo argomento consente di creare un nuovo gruppo di nodi, di migrare correttamente le applicazioni esistenti al nuovo gruppo e di rimuovere il precedente gruppo di nodi dal cluster. È possibile eseguire la migrazione a un nuovo gruppo di nodi utilizzando `eksctl` o la Console di gestione AWS.

- the section called "`eksctl`"
- the section called "Console di gestione AWS and AWS CLI"

### `eksctl`

Esegui la migrazione delle applicazioni a un nuovo gruppo di nodi con **`eksctl`**

Per ulteriori informazioni sull'utilizzo di eksctl per la migrazione, consulta Unmanaged nodegroups nella documentazione di `eksctl`.

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

> **Note**
>
> Questa procedura funziona solo per i cluster e per i gruppi di nodi creati con `eksctl`.

1. Recuperare il nome dei gruppi di nodi esistenti, sostituendo *`my-cluster`* con il nome del cluster.

```
eksctl get nodegroups --cluster=my-cluster
```

Di seguito viene riportato un output di esempio:

```
CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
 DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
               t3.medium         ami-05a71d034119ffc12
```

2. Avvia un nuovo gruppo di nodi con `eksctl` con il comando seguente. Nel comando, sostituire ogni *`example value`* con i propri valori. Il numero di versione non può essere successivo alla versione Kubernetes per il piano di controllo. Inoltre, non può essere più di due versioni secondarie precedenti rispetto a quella Kubernetes per il piano di controllo. Si consiglia di utilizzare la stessa versione del piano di controllo.

   Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:

   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

     Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

     Per bloccare l'accesso del pod a IMDS, aggiungere l'opzione `--disable-pod-imds` al seguente comando.

     > **Note**
     >
     > Per ulteriori bandiere disponibili e relative descrizioni, consulta https://eksctl.io/.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.36 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
```

```
  --nodes-max 4 \
  --managed=false
```

3. Quando il comando precedente viene completato, verificare che tutti i nodi abbiano raggiunto lo stato `Ready` con il comando seguente:

```
kubectl get nodes
```

4. Eliminare il gruppo di nodi originale con il comando seguente. Nel comando, sostituire ogni *`example value`* con i nomi dei cluster e dei gruppi di nodi:

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

Console di gestione AWS and AWS CLI

Migra le tue applicazioni in un nuovo gruppo di nodi con Console di gestione AWS e AWS CLI

1. Avvia un nuovo gruppo di nodi seguendo i passaggi descritti in Create self-managed Amazon Linux nodes.
2. Al termine della creazione dello stack, selezionalo nella console e scegli Output.
3. Registra il NodeInstanceRoleper il gruppo di nodi che è stato creato. Ciò è necessario per aggiungere i nuovi nodi Amazon EKS al cluster.

> **Note**
>
> Se si dispone di policy IAM aggiuntive associate al vecchio ruolo IAM del gruppo di nodi, associa le stesse policy al nuovo ruolo IAM del gruppo di nodi per mantenere tale funzionalità nel nuovo gruppo. Questo vale se hai aggiunto le autorizzazioni per il Cluster Autoscaler Kubernetes, ad esempio.

4. Aggiorna i gruppi di sicurezza per entrambi i gruppi di nodi in modo che possano comunicare tra loro. Per ulteriori informazioni, consulta the section called "Requisiti relativi al gruppo di sicurezza".
   a. Registra gli ID del gruppo di sicurezza per entrambi i gruppi dei nodi. Questo viene mostrato come NodeSecurityGroupvalore negli output dello AWS CloudFormation stack.

      È possibile utilizzare i seguenti comandi AWS CLI per ottenere gli ID dei gruppi di sicurezza dai nomi dello stack. In questi comandi, `oldNodes` è il nome AWS CloudFormation dello stack del

vecchio stack di nodi ed `newNodes` è il nome dello stack verso cui state migrando. Sostituisci ogni *`example value`* con i valori in tuo possesso.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
```

b. Aggiungi regole in ingresso per ciascun gruppo di sicurezza del nodo in modo da accettare il traffico tra loro.

   I seguenti comandi AWS CLI aggiungono regole in entrata a ciascun gruppo di sicurezza che consentono tutto il traffico su tutti i protocolli dell'altro gruppo di sicurezza. Questo consente ai pod in ogni gruppo di nodi di comunicare tra loro durante la migrazione del carico di lavoro verso il nuovo gruppo.

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. Modifica la configmap `aws-auth` per mappare il nuovo ruolo dell'istanza del nodo in RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Aggiungi una nuova voce `mapRoles` per il nuovo gruppo di nodi.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
```

```
        - system:nodes>
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

Sostituisci lo *ARN of instance role (not instance profile)* snippet con il NodeInstanceRolevalore registrato nel passaggio precedente. Quindi, salva e chiudi il file per applicare la configmap aggiornata.

6. Guarda lo stato dei nodi e attendi fino a quando i nuovi nodi si uniscono al cluster e raggiungono lo stato `Ready`.

```
kubectl get nodes --watch
```

7. (Facoltativo) Se stai utilizzando il Cluster Autoscaler Kubernetes, dimensiona l'implementazione fino a zero (0) repliche per evitare azioni di dimensionamento conflittuali.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. Utilizzare il comando seguente per il taint di ciascuno dei nodi che si desidera rimuovere con `NoSchedule`. In questo modo i nuovi pod non vengono pianificati o riprogrammati sui nodi che stai sostituendo. Per ulteriori informazioni, consulta Taints and Tolerations nella documentazione di Kubernetes.

```
kubectl taint nodes node_name key=value:NoSchedule
```

Se aggiorni i nodi a una nuova versione di Kubernetes, puoi identificare ed eseguire il taint di tutti i nodi di una determinata versione di Kubernetes (in questo caso, `1.34`) con il seguente frammento di codice. Il numero di versione non può essere successivo alla versione Kubernetes del piano di controllo. Inoltre, non può essere più di due versioni secondarie precedenti rispetto alla versione Kubernetes del piano di controllo. Si consiglia di utilizzare la stessa versione del piano di controllo.

```
K8S_VERSION=1.34
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
```

```
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. Determina il provider DNS del cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

Di seguito viene riportato un output di esempio. Questo cluster utilizza CoreDNS per la risoluzione DNS, ma il cluster può invece restituire `kube-dns`):

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10. Se l'implementazione corrente è in esecuzione per un numero di volte inferiore a 2 repliche, scalare la distribuzione a 2 repliche. Sostituire *`coredns`* con `kubedns` se l'output del comando precedente ha avuto tale risultato.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. Svuotare ciascuno dei nodi che si desidera rimuovere dal cluster con il comando seguente:

```
kubectl drain node_name --ignore-daemonsets --delete-emptydir-data
```

Se stai aggiornando i tuoi nodi a una nuova versione di Kubernetes, identifica e svuota tutti i nodi di una particolare versione di Kubernetes (in questo caso) con il seguente frammento di codice. *`1.34`*

```
K8S_VERSION=1.34
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-emptydir-data
done
```

12. Una volta terminata tale operazione, revocare le regole in ingresso del gruppo di sicurezza autorizzate in precedenza. Quindi, elimina lo stack per terminare le istanze. AWS CloudFormation

> **Note**
>
> Se hai collegato politiche IAM aggiuntive al tuo vecchio ruolo IAM del gruppo di nodi, ad esempio aggiungendo autorizzazioni per Kubernetes Cluster Autoscaler, scollega quelle politiche aggiuntive dal ruolo prima di poter eliminare lo stack. AWS CloudFormation

a. Revocare le regole in ingresso create in precedenza per i gruppi di sicurezza dei nodi. In questi comandi, `oldNodes` c'è il nome dello AWS CloudFormation stack del tuo vecchio stack di nodi ed `newNodes` è il nome dello stack verso cui stai migrando.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. Apri la AWS CloudFormation console.

c. Selezionare la pila del nodo precedente.

d. Scegli Elimina.

e. Nella finestra di dialogo di conferma Delete stack (Elimina stack) scegliere Delete stack.(Elimina stack).

13 Modifica il configmap `aws-auth` per rimuovere il precedente ruolo dell'istanza del nodo da RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Eliminare la voce `mapRoles` per il precedente gruppo di nodo.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

Salvare e chiudere il file per applicare la configmap aggiornata.

14. (Facoltativo) Se si sta utilizzando il Cluster Autoscaler Kubernetes, dimensionare l'implementazione a 1 replica.

> **Note**
>
> È inoltre necessario applicare i tag al nuovo gruppo Auto Scaling in modo appropriato (ad esempio, `k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/my-cluster`) e aggiornare il comando di implementazione di Cluster Autoscaler per puntare al nuovo gruppo Auto Scaling contrassegnato. Per ulteriori informazioni, consulta Cluster Autoscaler on. AWS

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (Facoltativo) Verifica se stai utilizzando la versione più recente del plug-in CNI di Amazon VPC per Kubernetes. Potrebbe essere necessario aggiornare la versione CNI per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

16. Se il cluster utilizza `kube-dns` per la risoluzione DNS (vedi [migrate-determine-dns-step]), riduci orizzontalmente l'implementazione `kube-dns` a 1 replica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

## Aggiorna un AWS CloudFormation stack di nodi

Questo argomento descrive come aggiornare uno stack di nodi AWS CloudFormation autogestito esistente con una nuova AMI. È possibile utilizzare questa procedura per aggiornare i nodi a una nuova versione di Kubernetes in seguito all'aggiornamento di un cluster. In caso contrario, è possibile eseguire l'aggiornamento all'AMI ottimizzata per Amazon EKS più recente per una versione Kubernetes esistente.

> **Important**
>
> In questo argomento vengono descritti gli aggiornamenti dei nodi per i gruppi di nodi autogestiti. Per informazioni sull'utilizzo di Simplify node lifecycle con gruppi di nodi gestiti, consulta the section called "Aggiorna".

L'ultimo AWS CloudFormation modello di nodo Amazon EKS predefinito è configurato per avviare un'istanza con la nuova AMI nel cluster prima di rimuoverne una vecchia, una alla volta. Questa configurazione garantisce sempre il conteggio desiderato del gruppo Auto Scaling delle istanze attive nel cluster durante l'aggiornamento in sequenza.

> **Note**
>
> Questo metodo non è supportato per i gruppi di nodi creati con `eksctl`. Se è stato creato il cluster o il gruppo di nodi con `eksctl`, consultare the section called "Migrazione".

1. Determinare il provider DNS del cluster.

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   Di seguito viene riportato un output di esempio. Questo cluster utilizza CoreDNS per la risoluzione DNS, ma il cluster può invece restituire `kube-dns`. L'output potrebbe avere un aspetto diverso a seconda della versione di `kubectl` che stai utilizzando.

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

2. Se l'implementazione corrente è in esecuzione per un numero di volte inferiore a 2 repliche, scalare la distribuzione a 2 repliche. Sostituire *coredns* con `kube-dns` se l'output del comando precedente ha avuto tale risultato.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

3. (Facoltativo) Se si sta utilizzando il Cluster Autoscaler Kubernetes, dimensiona l'implementazione fino a zero (0) repliche per evitare azioni di dimensionamento conflittuali.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. Determina il tipo di istanza e il numero di istanze desiderato del tuo attuale gruppo di nodi. Questi valori vengono immessi in un secondo momento quando si aggiorna il AWS CloudFormation modello per il gruppo.
   a. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
   b. Nel pannello di navigazione a sinistra, scegli Launch Configurations (Configurazioni di avvio) e prendi nota del tipo di istanza per la configurazione di avvio del nodo esistente.
   c. Nel pannello di navigazione a sinistra, scegli Auto Scaling Groups (Gruppi Auto Scaling) e prendi nota del conteggio delle istanze Desired (Desiderato) per il gruppo Auto Scaling del nodo.
5. Apri la AWS CloudFormation console.
6. Selezionare la pila del gruppo di nodi, quindi scegliere Aggiorna.
7. Selezionare Replace current template (Sostituisci modello corrente) e scegliere Amazon S3 URL (URL Amazon S3).
8. Per l'URL di  Amazon S3, incolla il seguente URL nell'area di testo per assicurarti di utilizzare la versione più recente del modello di nodo AWS CloudFormation . Quindi scegliere Next (Successivo):

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. Nella pagina Specify stack details (Specifica dettagli pila), compilare i parametri seguenti e scegliere Next (Successivo):

- NodeAutoScalingGroupDesiredCapacity— Inserisci il numero di istanze desiderato che hai registrato in un passaggio ??? precedente. In alternativa, inserire il nuovo numero desiderato di nodi come riferimento del dimensionamento quando viene aggiornata la pila.
- NodeAutoScalingGroupMaxSize— Inserite il numero massimo di nodi rispetto ai quali il gruppo Auto Scaling del nodo può essere ridimensionato. Questo valore deve essere almeno un nodo in più rispetto alla capacità desiderata. Ciò consente di eseguire un aggiornamento in sequenza dei nodi senza ridurre il numero di nodi durante l'aggiornamento.
- NodeInstanceType— Scegli il tipo di istanza che hai registrato in un passaggio ??? precedente. In alternativa, scegliere un tipo di istanza diverso per i nodi. Prima di scegliere un tipo di istanza diverso, esamina Choose an optimal Amazon EC2 node instance type. Ogni tipo di istanza Amazon EC2 supporta un numero massimo di interfacce di rete elastiche (interfaccia di rete) e ogni interfaccia di rete supporta un numero massimo di indirizzi IP. Poiché a ogni nodo di lavoro e Pod viene assegnato il proprio indirizzo IP, è importante scegliere un tipo di istanza che supporti il numero massimo di Pod che desideri eseguire su ogni nodo Amazon EC2. Per un elenco del numero di interfacce di rete e di indirizzi IP supportati dai tipi di istanza, consulta Indirizzi IP per interfaccia di rete e per tipo di istanza. Ad esempio, il tipo di istanza `m5.large` supporta un massimo di 30 indirizzi IP per il nodo (worker) e per i pod.

  > **Note**
  >
  > I tipi di istanza supportati per la versione più recente del plug-in CNI di Amazon VPC per Kubernetes sono riportati in vpc_ip_resource_limit.go su GitHub. Potrebbe essere necessario aggiornare la versione del plugin CNI di Amazon VPC per Kubernetes per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

  > **Important**
  >
  > Alcuni tipi di istanza potrebbero non essere disponibili in tutte le AWS regioni.

- NodeImageIdSSMParam— Il parametro Amazon EC2 Systems Manager dell'ID AMI a cui desideri eseguire l'aggiornamento. Il valore seguente utilizza l'AMI ottimizzata per Amazon EKS più recente per Kubernetes versione `1.36`.

  ```
  /aws/service/eks/optimized-ami/1.36/amazon-linux-2/recommended/image_id
  ```

Puoi sostituirlo *1.36* con una versione della piattaforma identica. In alternativa, può essere fino a una versione precedente alla versione di Kubernetes in esecuzione sul piano di controllo. Si consiglia di mantenere i nodi alla stessa versione del piano di controllo. Puoi anche sostituirla *amazon-linux-2* con un tipo di AMI diverso. Per ulteriori informazioni, consulta the section called "Ottieni le ultime novità IDs".

> **Note**
>
> L'utilizzo del parametro Amazon EC2 Systems Manager consente di aggiornare i nodi in futuro senza dover cercare e specificare un ID AMI. Se il tuo AWS CloudFormation stack utilizza questo valore, qualsiasi aggiornamento dello stack avvia sempre l'ultima AMI ottimizzata Amazon EKS consigliata per la versione di Kubernetes specificata. Questo è il caso anche se non si modificano valori nel modello.

- NodeImageId— Per utilizzare la tua AMI personalizzata, inserisci l'ID dell'AMI da utilizzare.

> **Important**
>
> Questo valore ha la precedenza su qualsiasi valore specificato per. NodeImageIdSSMParam Se desideri utilizzare il NodeImageIdSSMParam valore, assicurati che il valore per NodeImageId sia vuoto.

- DisabilitaIMDSv1: ogni nodo supporta Instance Metadata Service versione 1 (IMDSv1) e IMDSv2 per impostazione predefinita. Tuttavia, è possibile disabilitare IMDSv1. Seleziona vero se non si desidera che nessun nodo o pod pianificato sul gruppo di nodi utilizzi IMDSv1. Per ulteriori informazioni su IMDS, consulta Configurazione del servizio di metadati dell'istanza. Se hai implementato i ruoli IAM per gli account di servizio, assegna le autorizzazioni necessarie direttamente a tutti i Pod che richiedono l'accesso ai AWS servizi. In questo modo, nessun Pod del cluster richiede l'accesso all'IMDS per altri motivi, come il recupero della regione corrente. AWS Quindi, è anche possibile disabilitare l'accesso a IMDSv2 per i pod che non utilizzano la rete host. Per ulteriori informazioni, consulta Limita l'accesso al profilo di istanza assegnato al nodo (worker).

10. (Facoltativo) Nella pagina Options (Opzioni), contrassegna con dei tag le risorse della pila. Scegli Avanti.
11. Nella pagina Verifica, esaminare le informazioni, confermare che la pila è in grado di creare risorse IAM, quindi scegliere Aggiorna pila.

> **Note**
>
> L'aggiornamento di ogni nodo nel cluster richiede diversi minuti. Attendi il completamento dell'aggiornamento di tutti i nodi prima di eseguire la procedura successiva.

12. Se il provider DNS del cluster è `kube-dns`, riduci orizzontalmente l'implementazione di `kube-dns` a una replica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (Facoltativo) Se si sta utilizzando il Cluster Autoscaler Kubernetes, dimensionare l'implementazione al numero di repliche desiderato.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (Facoltativo) Verifica che si sta utilizzando la versione più recente del plugin CNI di Amazon VPC per Kubernetes. Potrebbe essere necessario aggiornare la versione del plugin CNI di Amazon VPC per Kubernetes per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

# Semplifica la gestione dell'elaborazione con AWS Fargate

Questo argomento illustra l'utilizzo di Amazon EKS per eseguire Kubernetes Pods su Fargate. AWS Fargate è una tecnologia che fornisce capacità di calcolo on demand e di dimensioni adeguate per i container. Con Fargate, non devi effettuare personalmente il provisioning, la configurazione o il dimensionamento dei gruppi di macchine virtuali per eseguire i container. Non devi inoltre scegliere i tipi di server, decidere quando dimensionare i gruppi di nodi oppure ottimizzare l'impacchettamento dei cluster.

Puoi controllare quali pod sono avviati su Fargate e come vengono eseguiti con profili Fargate. I profili Fargate sono definiti come parte del cluster Amazon EKS. Amazon EKS integra Kubernetes con Fargate utilizzando controller creati utilizzando il modello upstream ed estensibile fornito da Kubernetes. AWS Questi controller vengono eseguiti come parte del piano di controllo di Kubernetes gestito da Amazon EKS e sono responsabili della pianificazione dei pod Kubernetes nativo su Fargate. I controller Fargate includono un nuovo pianificazione che viene eseguito insieme al pianificatore di default di Kubernetes, oltre a diversi controller di ammissione mutanti e convalidanti.

Quando avvii un pod che soddisfa i criteri per l'esecuzione su Fargate, i controller Fargate in esecuzione nel cluster riconoscono, aggiornano e pianificano il pod su Fargate.

Questo argomento descrive i diversi componenti di pod in esecuzione su Fargate, e illustra considerazioni speciali per l'utilizzo di Fargate con Amazon EKS.

## AWS Considerazioni su Fargate

Di seguito sono elencati alcuni punti da considerare sull'uso di Fargate in Amazon EKS.

- Ogni pod in esecuzione su Fargate ha un proprio limite di calcolo. Non condividono il kernel sottostante, le risorse CPU, le risorse di memoria o l'interfaccia di rete elastica con un altro pod.
- I Network Load Balancer e gli Application Load Balancer (ALB) possono essere utilizzati solo con Fargate con destinazioni IP. Per ulteriori informazioni, consulta the section called "Creazione di un Network Load Balancer" e the section called "Bilanciamento del carico di applicazione".
- I servizi esposti Fargate vengono eseguiti solo in modalità IP di tipo destinazione e non in modalità IP del nodo. Il modo consigliato per verificare la connettività da un servizio in esecuzione su un nodo gestito e da un servizio in esecuzione su Fargate è quello di connettersi tramite il nome del servizio.
- Nel momento in cui sono programmati per l'esecuzione su Fargate, i pod devono corrispondere a un profilo Fargate. I pod che non corrispondono a un profilo Fargate potrebbero rimanere bloccati nello stato `Pending`. Se esiste un profilo Fargate corrispondente, puoi eliminare i pod in sospeso creati per riprogrammarli su Fargate.
- I Daemonset non sono supportati su Fargate. Se l'applicazione richiede un daemon, riconfigura quel daemon per essere eseguito nei pod come container sidecar.
- I container privilegiati non sono supportati su Fargate.
- I pod in esecuzione su Fargate non possono specificare `HostPort` o `HostNetwork` nel manifesto del pod.
- Per i pod Fargate, il valore di default per il limite soft `nofile` e `nproc` è 1.024, mentre per il limite hard è 65.535.
- Le GPU non sono attualmente disponibili su Fargate.
- I pod eseguiti su Fargate sono supportati solo su sottoreti private (con accesso tramite gateway NAT ai AWS servizi, ma non un percorso diretto verso un gateway Internet), quindi il VPC del cluster deve disporre di sottoreti private disponibili. Per i cluster senza accesso Internet in uscita, consulta the section called "Cluster privati".

- Puoi utilizzare Adjust pod resources with Vertical Pod Autoscaler per impostare la dimensione corretta iniziale della CPU e la memoria per i Pod Fargate, quindi utilizza Scale pod deployments with Horizontal Pod Autoscaler per scalare quei Pod. Se desideri che il Vertical Pod Autoscaler ridistribuisca automaticamente i pod su Fargate con combinazioni di CPU e memoria più grandi, imposta la modalità per il Vertical Pod Autoscaler su `Auto` o `Recreate` r per garantire la corretta funzionalità. Per ulteriori informazioni, consulta la documentazione di Vertical Pod Autoscaler su GitHub.
- La risoluzione DNS e i nomi host DNS devono essere abilitati per il VPC. Per ulteriori informazioni, consulta Visualizzazione e aggiornamento del supporto DNS per il VPC.
- Amazon EKS Fargate aggiunge una difesa approfondita per le applicazioni Kubernetes isolando ogni pod all'interno di una macchina virtuale (VM). Questo limite VM impedisce l'accesso alle risorse basate su host utilizzate da altri pod in caso di evasione da un container, un metodo comune per attaccare le applicazioni containerizzate e ottenere l'accesso a risorse esterne al container.

  L'utilizzo di Amazon EKS non modifica le tue responsabilità ai sensi del modello di responsabilità condivisa. È necessario considerare attentamente la configurazione dei controlli di sicurezza e gestione del cluster. Il modo più sicuro per isolare un'applicazione è sempre quello di eseguirla in un cluster separato.
- I profili Fargate supportano la specificazione di sottoreti da blocchi CIDR secondari del VPC. È possibile specificare un blocco CIDR secondario. Ciò è necessario perché in una sottorete è disponibile un numero limitato di indirizzi IP. Di conseguenza, nel cluster puoi creare un numero limitato di pod. L'uso di sottoreti diverse per i pod ti consente di aumentare il numero di indirizzi IP disponibili. Per ulteriori informazioni, consulta Aggiunta di blocchi CIDR IPv4 a un VPC.
- Il servizio di metadati di istanza (IMDS) Amazon EC2 non è disponibile per i pod distribuiti ai nodi Fargate. Se disponi di pod distribuiti in Fargate che necessitano di credenziali IAM, assegnali ai pod utilizzando i ruoli IAM per gli account di servizio. Se i pod hanno bisogno di accedere ad altre informazioni disponibili tramite IMDS, devi eseguire la codifica fissa di queste informazioni nelle specifiche del pod. Ciò include la AWS regione o la zona di disponibilità in cui viene distribuito un pod.
- Non puoi distribuire i Fargate Pod negli AWS avamposti, nelle AWS lunghezze d'onda o nelle zone locali. AWS
- Periodicamente, Amazon EKS deve applicare patch ai Pod Fargate per garantire che siano sicuri. Gli aggiornamenti vengono effettuati in modo da avere il minor impatto possibile. Tuttavia, se i pod non vengono espulsi con successo, a volte è necessario eliminarli. Di seguito sono riportate le

azioni che è possibile intraprendere per ridurre al minimo le interruzioni. Per ulteriori informazioni, consulta the section called "Eventi di applicazione di patch del sistema operativo".

- Il plug-in CNI di Amazon VPC per Amazon EKS è installato sui nodi Fargate. Non puoi utilizzare Alternate CNI plugins for Amazon EKS clusters con i nodi Fargate.
- Un pod in esecuzione su Fargate monta automaticamente un file system Amazon EFS senza bisogno della procedura di installazione manuale del driver. Non puoi utilizzare il provisioning dinamico dei volumi persistenti con nodi Fargate, ma puoi usare quello statico.
- Amazon EKS non supporta Fargate Spot.
- Non puoi montare volumi Amazon EBS sui pod Fargate.
- Puoi eseguire il controller Amazon EBS CSI sui nodi Fargate, ma il nodo Amazon EBS CSI può funzionare solo su istanze Amazon EC2. DaemonSet
- Rollback della versione: il rollback della versione non è supportato per i nodi di lavoro Fargate. È possibile ripristinare il piano di controllo di un cluster che utilizza Fargate, ma i pod Fargate che eseguono la stessa versione di Kubernetes del piano di controllo attivano una visualizzazione distorta della versione kubelet con lo stato ERROR. Se disponi di pod Fargate con una versione secondaria corrispondente alla versione del piano di controllo, eliminali prima di eseguire il rollback. Quindi ripristina il piano di controllo. Tutti i pod rimanenti vengono avviati con la versione ripristinata quando li ridistribuisci. In alternativa, usalo `--force` per bypassare l'insight check, ma procedere con una violazione della distorsione della versione di Kubelet potrebbe comportare un comportamento imprevisto. Per ulteriori informazioni, consulta the section called "Ripristina la versione di Kubernetes".
- Dopo che un Kubernetes Job è stato contrassegnato `Completed` o `Failed`, i pod creati normalmente continuano a esistere. Questo comportamento ti consente di visualizzare i tuoi log e risultati, ma con Fargate dovrai sostenere dei costi se non ripulisci il processo in seguito.

  Per eliminare automaticamente i pod correlati dopo il completamento o un errore di un processo, puoi specificare un periodo di tempo utilizzando il controller time-to-live (TTL). L'esempio seguente mostra la specifica di `.spec.ttlSecondsAfterFinished` nel manifesto del processo.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
```

```
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

## Tabella di confronto Fargate

| Criteri | AWS Fargate |
|---|---|
| Si può implementare in AWS Outposts | No |
| Si può implementare in una Zona locale AWS | No |
| Può eseguire container che richiedono Windows | No |
| Può eseguire container che richiedono Linux | Sì |
| Può eseguire carichi di lavoro che richiedono il chip Inferentia | No |
| Può eseguire carichi di lavoro che richiedono una GPU | No |
| Può eseguire carichi di lavoro che richiedono processori Arm | No |
| Può eseguire AWS Bottlerocket | No |
| I pod condividono un ambiente di runtime del kernel con altri pod | No: ogni pod ha un kernel dedicato |
| I pod condividono risorse di CPU, memoria, archiviazione e rete con altri pod. | No: ogni pod dispone di risorse dedicate e può essere ridimensionato in modo indipendente per massimizzare l'utilizzo delle risorse. |

| Criteri | AWS Fargate |
| --- | --- |
| I pod possono utilizzare più hardware e memoria rispetto alle specifiche dei pod | No: il pod può essere comunque reimpleme ntato utilizzando una vCPU e una configura zione di memoria più grandi. |
| È necessario implementare e gestire le istanze Amazon EC2 | No |
| È necessario proteggere, mantenere e applicare patch al sistema operativo delle istanze Amazon EC2 | No |
| Può fornire argomenti di bootstrap all'imple mentazione di un nodo, come argomenti kubelet aggiuntivi. | No |
| Può assegnare indirizzi IP ai pod da un intervallo CIDR diverso rispetto all'indirizzo IP assegnato al nodo. | No |
| Puoi eseguire SSH nel nodo | No: non esiste un sistema operativo host del nodo su cui eseguire il SSH. |
| Puoi implementare un'AMI personalizzata nei nodi | No |
| Può implementare una CNI personalizzata nei nodi | No |
| É necessario aggiornare l'AMI del nodo per conto proprio | No |
| È necessario aggiornare la versione del nodo Kubernetes per conto proprio | No: non sei tu a gestire i nodi. |
| Può utilizzare lo spazio di archiviazione Amazon EBS con i pod | No |

| Criteri | AWS Fargate |
| --- | --- |
| Può utilizzare lo spazio di archiviazione Amazon EFS con i pod | Sì |
| Può utilizzare lo spazio di archiviazione Amazon FSx per Lustre con i pod | No |
| Può utilizzare Network Load Balancer per i servizi | Sì, quando si utilizza Create a network load balancer |
| I pod possono essere eseguiti in una sottorete pubblica | No |
| Può assegnare diversi gruppi di sicurezza VPC a singoli pod | Sì |
| Può eseguire Kubernetes DaemonSets | No |
| Supporto `HostPort` e `HostNetwork` nel manifesto pod | No |
| AWS Disponibilità della regione | Alcune regioni supportate da Amazon EKS |
| Può eseguire container su host dedicati Amazon EC2 | No |
| Prezzi | Costo di una singola configurazione di memoria Fargate e CPU. Ogni pod ha il rispettivo costo. Per ulteriori informazioni, consulta Prezzi di Fargate AWS. |

## Inizia a usare AWS Fargate per il tuo cluster

Questo argomento descrive come iniziare a eseguire Pods su AWS Fargate con il tuo cluster Amazon EKS.

Se limiti l'accesso all'endpoint pubblico del cluster utilizzando blocchi CIDR, consigliamo di abilitare anche l'accesso agli endpoint privati. In questo modo i pod Fargate possono comunicare con il

cluster. Senza l'endpoint privato abilitato, i blocchi CIDR specificati per l'accesso pubblico devono includere le origini di uscita dal VPC. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

Prerequisito

Un cluster esistente. Se si dispone già di un cluster Amazon EKS, consultare la pagina Nozioni di base.

## Fase 1: assicurarsi che i nodi esistenti possano comunicare con i pod Fargate

Se si utilizza un nuovo cluster senza nodi o un cluster con solo gruppi di nodi gestiti (consultare the section called "Gruppi di nodi gestiti"), è possibile passare a the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate".

Supponiamo di lavorare con un cluster esistente che dispone già di nodi associati. Assicurarsi che i pod su questi nodi possano comunicare liberamente con i pod in esecuzione su Fargate. I pod in esecuzione su Fargate vengono configurati automaticamente per utilizzare il gruppo di sicurezza del cluster per il cluster a cui sono associati. È necessario assicurarsi che eventuali i nodi esistenti nel cluster possano inviare e ricevere traffico da e verso il gruppo di sicurezza del cluster. I gruppi di nodi gestiti vengono configurati automaticamente per utilizzare anche il gruppo di sicurezza del cluster, quindi non è necessario modificarli o verificare questa compatibilità (consultare the section called "Gruppi di nodi gestiti").

Per i gruppi di nodi esistenti che sono stati creati con `eksctl` o i AWS CloudFormation modelli gestiti di Amazon EKS, puoi aggiungere manualmente il gruppo di sicurezza del cluster ai nodi. In alternativa, è possibile modificare il modello di avvio del gruppo Auto Scaling per il gruppo di nodi per collegare il gruppo di sicurezza cluster alle istanze. Per ulteriori informazioni, consultare Changing an instance's security groups nella Guida per l'utente di Amazon VPC.

Puoi verificare la presenza di un gruppo di sicurezza per il Console di gestione AWS tuo cluster nella sezione Rete relativa al cluster. In alternativa, è possibile eseguire questa operazione utilizzando il seguente AWS comando CLI. Se utilizzi questo comando, sostituisci `<my-cluster>` con il nome del cluster.

```
aws eks describe-cluster --name <my-cluster> --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

## Fase 2: creare un ruolo di esecuzione del pod Fargate

Quando il cluster crea Pod su AWS Fargate, i componenti che girano sull'infrastruttura Fargate devono effettuare chiamate AWS APIs per conto dell'utente. Il ruolo di esecuzione del pod Amazon EKS fornisce le autorizzazioni IAM per eseguire questa operazione. Per creare un ruolo di esecuzione di AWS Fargate Pod, vedere. the section called "Ruolo IAM per l'esecuzione del pod"

> **Note**
>
> Se il cluster è stato creato con `eksctl` utilizzando l'opzione `--fargate`, dispone già di un ruolo di esecuzione del pod disponibile nella console IAM con il modello `eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL`. Allo stesso modo, se si utilizza `eksctl` per creare i profili Fargate, `eksctl` crea il ruolo di esecuzione del pod se non ne esiste già uno.

## Fase 3: creare un profilo Fargate per il cluster

Prima di poter pianificare i pod in esecuzione su Fargate nel cluster, sarà necessario definire un profilo Fargate che specifichi i pod che utilizzano Fargate al momento dell'avvio. Per ulteriori informazioni, consulta the section called "Definisci i profili".

> **Note**
>
> Se il cluster è stato creato con `eksctl` utilizzando l'opzione `--fargate`, allora un profilo Fargate è già stato creato per il cluster con selettori per tutti i pod nei namespace `kube-system` e `default`. Utilizza la procedura seguente per creare profili Fargate per gli altri namespace con cui desideri utilizzare Fargate.

È possibile creare un profilo Fargate utilizzando uno di questi due strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

**`eksctl`**

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

Per creare un profilo Fargate con **`eksctl`**

Crea il tuo profilo Fargate con il seguente comando `eksctl`, sostituendo ogni `<example value>` con i valori in tuo possesso. È necessario specificare un namespace. Tuttavia, l'opzione `--labels` non è obbligatoria.

```
eksctl create fargateprofile \
    --cluster <my-cluster> \
    --name <my-fargate-profile> \
    --namespace <my-kubernetes-namespace> \
    --labels <key=value>
```

È possibile utilizzare determinati caratteri jolly per `<my-kubernetes-namespace>` e etichette `<key=value>`. Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

Console di gestione AWS

Creazione di un profilo Fargate con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il cluster per cui creare un profilo Fargate.
3. Scegli la scheda Calcolo.
4. Nella sezione Profili Fargate, scegli Aggiungi profilo Fargate.
5. Nella pagina Configura il profilo Fargate, procedi come segue:
   a. In Nome, inserisci un nome per il profilo Fargate. Il nome deve essere univoco.
   b. Come Ruolo di esecuzione pod, scegliere il ruolo di esecuzione del pod da utilizzare con il profilo Fargate. Vengono visualizzati solo i ruoli IAM con il principale del servizio `eks-`

`fargate-pods.amazonaws.com`. Se non vedi alcun ruolo nell'elenco, è necessario crearne uno. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".

c. Modifica le sottoreti selezionate in base alle esigenze.

> **Note**
>
> Solo le sottoreti private sono supportate per i pod in esecuzione su Fargate.

d. In Tag, puoi facoltativamente aggiungere tag al tuo profilo Fargate. Questi tag non si propagano ad altre risorse associate al profilo come i pod.

e. Scegli Next (Successivo).

6. Nella pagina Configura selezione pod, procedi come segue:

a. In Namespace, inserisci un namespace che corrisponda ai pod.

- È possibile utilizzare namespace specifici da abbinare, ad esempio `kube-system` o `default`.
- È possibile utilizzare determinati caratteri jolly (ad esempio, `prod-*`) per abbinare più namespace (ad esempio, `prod-deployment` e `prod-test`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

b. (Facoltativo) Aggiungere etichette Kubernetes al selettore. In particolare, aggiungerle al selettore con cui i pod nel namespace specificato devono corrispondere.

- È possibile aggiungere l'etichetta `infrastructure: fargate` al selettore in modo che solo i pod nel namespace specificato che hanno anche l'etichetta Kubernetes `infrastructure: fargate` corrispondano al selettore.
- È possibile utilizzare determinati caratteri jolly (ad esempio, `key?: value?`) per abbinare più namespace (ad esempio, `keya: valuea` e `keyb: valueb`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

c. Scegli Next (Successivo).

7. Nella pagina Rivedi e crea, controlla le informazioni relative al profilo Fargate e scegli Crea.

## Fase 4: aggiornare CoreDNS

Per impostazione predefinita, CoredNS è configurato per l'esecuzione sull' EC2 infrastruttura Amazon su cluster Amazon EKS. Se si desidera solo eseguire i pod su Fargate nel cluster, seguire la procedura riportata di seguito.

> **Note**
>
> Se è stato creato il cluster `eksctl` utilizzando l'opzione `--fargate` è possibile passare a the section called "Fasi successive".

1. Creare un profilo Fargate per CoreDNS con il seguente comando. Sostituiscilo `<my-cluster>` con il nome del cluster, `<111122223333>` con l'ID dell'account, `<AmazonEKSFargatePodExecutionRole>` con il nome del ruolo di esecuzione del Pod e `<000000000000000a>` `<000000000000000c>` con le sottoreti IDs private. `<000000000000000b>` Se non si dispone di un ruolo di esecuzione del pod, è necessario prima crearne uno (consultare the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate").

   > **Important**
   >
   > L'ARN del ruolo non può includere un percorso diverso da `/`. Ad esempio, se il nome del ruolo è `development/apps/AmazonEKSFargatePodExecutionRole`, è necessario modificarlo in `AmazonEKSFargatePodExecutionRole` quando si specifica l'ARN per tale ruolo. Il formato dell'ARN del ruolo deve essere `arn:aws:iam::<111122223333>:role/<AmazonEKSFargatePodExecutionRole>`.

   ```
   aws eks create-fargate-profile \
       --fargate-profile-name coredns \
       --cluster-name <my-cluster> \
       --pod-execution-role-arn arn:aws: iam::<111122223333>:role/
   <AmazonEKSFargatePodExecutionRole> \
       --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
       --subnets subnet-<000000000000000a> subnet-<000000000000000b> subnet-
   <000000000000000c>
   ```

2. Attivare un rollout dell'implementazione `coredns`.

   ```
   kubectl rollout restart -n kube-system deployment coredns
   ```

### Fasi successive

- Puoi avviare la migrazione delle applicazioni esistenti per l'esecuzione su Fargate con il seguente flusso di lavoro.
  a. the section called "Creazione di un profilo Fargate" che corrisponda al namespace Kubernetes e alle etichette Kubernetes dell'applicazione.
  b. Eliminare e creare nuovamente tutti i pod esistenti in modo che siano programmati su Fargate. Modificare `<namespace>` e `<deployment-type>` per aggiornare i pod specifici.

```
kubectl rollout restart -n <namespace> deployment <deployment-type>
```

- Implementare the section called "Bilanciamento del carico di applicazione" per permettere l'esecuzione di oggetti Ingress per i pod in esecuzione su Fargate.
- È possibile utilizzare the section called "Vertical Pod Autoscaler" per impostare la dimensione corretta iniziale della CPU e la memoria per i pod Fargate, quindi utilizzare il comando the section called "Horizontal Pod Autoscaler" per scalare tali pod. Se si desidera che il Vertical Pod Autoscaler implementi di nuovo automaticamente i pod su Fargate con combinazioni di CPU e memoria più grandi, impostare la modalità del Vertical Pod Autoscaler su `Auto` o `Recreate`. Ciò garantisce una corretta funzionalità. Per ulteriori informazioni, consulta la documentazione di Vertical Pod Autoscaler su GitHub.
- Ora puoi impostare il collector AWS Distro per OpenTelemetry (ADOT) per il monitoraggio delle applicazioni seguendo queste istruzioni.

## Definisci quali pod utilizzano AWS Fargate durante l'avvio

Prima di programmare i pod su Fargate nel cluster, devi definire almeno un profilo Fargate che specifichi i pod utilizzati da Fargate al momento dell'avvio.

Come amministratore, puoi utilizzare il profilo Fargate per dichiarare quali pod vengono eseguiti su Fargate. Puoi farlo attraverso i selettori del profilo. Puoi aggiungere fino a cinque selettori a ogni profilo. Ogni selettore deve contenere uno spazio dei nomi. Il selettore può includere anche etichette. Il campo etichetta è costituito da più coppie chiave-valore facoltative. I pod che corrispondono ai selettori sono programmati su Fargate. I pod vengono abbinati utilizzando uno spazio dei nomi e le etichette specificate nel selettore. Se un selettore di namespace è definito senza etichette, Amazon EKS tenterà di pianificare tutti i pod che vengono eseguiti in tale namespace su Fargate utilizzando il profilo. Se un pod da programmare corrisponde a uno dei selettori nel profilo Fargate, allora quel pod viene programmato su Fargate.

Se un pod corrisponde a più profili Fargate, puoi specificare quale profilo viene utilizzato dal pod aggiungendo la seguente etichetta Kubernetes alla specifica del pod: `eks.amazonaws.com/fargate-profile: my-fargate-profile`. Il pod deve corrispondere a un selettore in quel profilo per essere pianificato su Fargate. Le regole di affinità/anti-affinità Kubernetes non si applicano e non sono necessarie con i pod Fargate di Amazon EKS.

Quando crei un profilo Fargate, devi specificare il ruolo di esecuzione di un pod. Questo ruolo di esecuzione è per i componenti Amazon EKS che vengono eseguiti su infrastruttura Fargate utilizzando il profilo. Viene aggiunto al controllo di accesso basato sul ruolo (RBAC) del cluster di Kubernetes per l'autorizzazione. Ciò consente a `kubelet` in esecuzione sull'infrastruttura Fargate di registrarsi con il cluster Amazon EKS in modo che possa essere visualizzato nel cluster come nodo. Il ruolo di esecuzione del pod fornisce anche le autorizzazioni IAM per l'infrastruttura Fargate per consentire l'accesso in lettura ai repository di immagini di Amazon ECR. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".

I profili Fargate non possono essere modificati. Tuttavia, è possibile creare un nuovo profilo aggiornato per sostituire un profilo esistente e quindi eliminare l'originale.

> **Note**
>
> Qualsiasi pod in esecuzione utilizzando un profilo Fargate viene arrestato e messo in sospeso quando il profilo viene eliminato.

Se tutti i profili in un cluster sono nello stato `DELETING`, è necessario attendere che tale profilo Fargate finisca l'eliminazione prima di poter creare altri profili in tale cluster.

> **Note**
>
> Attualmente, Fargate non supporta topologySpreadConstraints di Kubernetes.

Amazon EKS e Fargate cercano di implementare i pod su ciascuna delle sottoreti definite nel profilo Fargate. Tuttavia, potresti ritrovarti con una diffusione irregolare. Se devi avere una diffusione uniforme, usa due profili Fargate. Anche la diffusione è importante in scenari in cui desideri implementare due repliche e non vuoi tempi di inattività. È consigliabile che ogni profilo abbia una sola sottorete.

## Componenti del profilo Fargate

Un profilo Fargate contiene i componenti elencati di seguito.

Ruolo di esecuzione del pod

Quando il cluster crea pod su AWS Fargate, il `kubelet` in esecuzione sull'infrastruttura di Fargate deve effettuare chiamate ad API AWS per tuo conto. Ad esempio, deve effettuare chiamate per estrarre le immagini del container da Amazon ECR. Il ruolo di esecuzione del pod Amazon EKS fornisce le autorizzazioni IAM per eseguire questa operazione.

Quando crei un profilo Fargate, devi specificare il ruolo di esecuzione di un pod da utilizzare col tuo pod. Questo ruolo viene aggiunto al controllo di accesso basato sul ruolo (RBAC) del cluster di Kubernetes per l'autorizzazione. Ciò consente a `kubelet` in esecuzione sull'infrastruttura Fargate di registrarsi con il cluster Amazon EKS in modo che possa essere visualizzato nel cluster come nodo. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".

Sottoreti

Gli ID delle sottoreti per avviare i pod che utilizzano questo profilo. A questo momento, ai pod in esecuzione su Fargate non vengono assegnati indirizzi IP pubblici. Di conseguenza, in virtù di questo parametro, sono accettate solo le sottoreti private senza routing diretto a un gateway Internet.

Selettori

I selettori da abbinare ai pod per utilizzare questo profilo Fargate. È possibile specificare fino a cinque selettori in un profilo Fargate. I selettori hanno le seguenti componenti:

- Spazio dei nomi: specifica uno spazio dei nomi per un selettore. Il selettore corrisponde solo ai pod che vengono creati in questo namespace. Tuttavia, è possibile creare più selettori per rivolgersi a più spazi dei nomi.
- Etichette: è possibile specificare facoltativamente le etichette Kubernetes che corrispondono al selettore. Il selettore corrisponde solo ai pod che hanno tutte le etichette specificate nel selettore.

## Wildcard del profilo di Fargate

Oltre ai caratteri consentiti da Kubernetes, puoi utilizzare `*` e `?` nei criteri di selezione per namespace, chiavi di etichette e valori di etichette:

- `*` rappresenta nessuno, uno o più caratteri. Ad esempio, `prod*` può rappresentare `prod` e `prod-metrics`.
- `?` rappresenta un singolo carattere (ad esempio, `value?` può rappresentare `valuea`). Tuttavia, non può rappresentare `value` e `value-a`, perché `?` può rappresentare solo un carattere.

Questi caratteri jolly possono essere usati in qualsiasi posizione e in combinazione (ad esempio, `prod*`, `*dev`, e `frontend*?`). Altri caratteri jolly e forme di corrispondenza del modello, come le espressioni regolari, non sono supportati.

Se sono presenti più profili corrispondenti per il namespace e le etichette nelle specifiche del pod, Fargate seleziona il profilo in base all'ordinamento alfanumerico e al nome del profilo. Ad esempio, se entrambi i profili A (con il nome `beta-workload`) e il profilo B (con il nome `prod-workload`) hanno dei selettori corrispondenti per i pod da avviare, Fargate sceglie il profilo A (`beta-workload`) per i pod. I pod hanno etichette con il profilo A sui pod (ad esempio, `eks.amazonaws.com/fargate-profile=beta-workload`).

Se desideri eseguire la migrazione dei pod Fargate esistenti su nuovi profili che utilizzano caratteri jolly, puoi farlo in due modi:

- Crea un nuovo profilo con i selettori corrispondenti, quindi elimina i vecchi profili. I pod etichettati con vecchi profili vengono riprogrammati con nuovi profili corrispondenti.
- Se desideri eseguire la migrazione dei carichi di lavoro ma non hai la certezza di quali siano le etichette Fargate su ogni pod Fargate, puoi utilizzare il seguente metodo. Crea un nuovo profilo con un nome che ordina in ordine alfanumerico per primo tra i profili dello stesso cluster. Quindi, ricicla i pod Fargate che devono essere migrati a nuovi profili.

## Creazione di un profilo Fargate

Questa sezione descrive come creare un profilo Fargate. Devi inoltre aver creato un ruolo di esecuzione del pod da utilizzare per il profilo Fargate. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod". I pod in esecuzione su Fargate sono supportati solo su sottoreti private con un accesso NAT gateway ai servizi AWS, ma senza una route diretta a un gateway Internet. In questo modo il VPC del cluster deve avere sottoreti private disponibili.

Puoi creare un profilo con ciò che segue:

- the section called "`eksctl`"

- the section called "Console di gestione AWS"

### `eksctl`

Creazione di un profilo Fargate con **`eksctl`**

Crea il tuo profilo Fargate con il seguente comando `eksctl`, sostituendo ogni valore di esempio con i valori in tuo possesso. Devi specificare un namespace. Tuttavia, l'opzione `--labels` non è obbligatoria.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

È possibile utilizzare determinati caratteri jolly per `my-kubernetes-namespace` e etichette `key=value`. Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

### Console di gestione AWS

Creazione di un profilo Fargate con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il cluster per cui creare un profilo Fargate.
3. Scegli la scheda Calcolo.
4. Nella sezione Profili Fargate, scegli Aggiungi profilo Fargate.
5. Nella pagina Configure Fargate profile (Configura profilo Fargate), procedere come segue:
   a. In Nome, inserisci un nome univoco per il profilo Fargate, ad esempio `my-profile`.
   b. Per il ruolo di esecuzione del pod, scegli il ruolo di esecuzione del pod da utilizzare con il profilo Fargate. Vengono visualizzati solo i ruoli IAM con il principale del servizio `eks-fargate-pods.amazonaws.com`. Se non vedi alcun ruolo nell'elenco, è necessario crearne uno. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".
   c. Modifica le sottoreti selezionate in base alle esigenze.

> **Note**
>
> Solo le sottoreti private sono supportate per i pod in esecuzione su Fargate.

d. In Tag, puoi facoltativamente aggiungere tag al tuo profilo Fargate. Questi tag non si propagano ad altre risorse associate al profilo, ad esempio i suoi pod.

e. Scegli Next (Successivo).

6. Nella pagina Configura selezione pod, procedi come segue:

a. In Namespace, inserisci uno spazio dei nomi che corrisponda ai pod.

- È possibile utilizzare spazi dei nomi specifici da abbinare, ad esempio `kube-system` o `default`.
- È possibile utilizzare determinati caratteri jolly (ad esempio, `prod-*`) per abbinare più spazi dei nomi (ad esempio, `prod-deployment` e `prod-test`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

b. (Facoltativo) Aggiungere etichette Kubernetes al selettore. In particolare, aggiungile al selettore a cui devono corrispondere i pod nel namespace specificato.

- Puoi aggiungere l'etichetta `infrastructure: fargate` al selettore in modo che solo i pod nel namespace specificato che hanno anche l'etichetta Kubernetes `infrastructure: fargate` corrispondano al selettore.
- È possibile utilizzare determinati caratteri jolly (ad esempio, `key?: value?`) per abbinare più spazi dei nomi (ad esempio, `keya: valuea` e `keyb: valueb`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

c. Scegli Next (Successivo).

7. Nella pagina Rivedi e crea, controlla le informazioni relative al profilo Fargate e scegli Crea.

## Eliminare un profilo Fargate

Questo argomento descrive come eliminare un profilo Fargate. Quando si elimina un profilo Fargate, eventuali pod pianificati su Fargate con tale profilo vengono eliminati. Se tali pod corrispondono a un altro profilo Fargate, vengono pianificati su Fargate con quel profilo. Se non corrispondono più a nessun profilo Fargate, non saranno pianificati su Fargate e potrebbero rimanere in sospeso.

Solo un profilo Fargate in un cluster alla volta può essere nello stato `DELETING`. Devi attendere il completamento dell'eliminazione di un profilo Fargate prima di poter eliminare altri profili nel cluster.

# possibile eliminare un profilo con uno qualsiasi dei seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "CLI AWS"

## `eksctl`

Eliminare un profilo Fargate con **`eksctl`**

Utilizza il comando seguente per eliminare un profilo da un cluster. Sostituire ogni *`valore di esempio`* con i propri valori.

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

## Console di gestione AWS

Eliminare un profilo Fargate con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli il cluster da cui desideri eliminare il profilo Fargate.
3. Scegli la scheda Calcolo.
4. Scegli il profilo Fargate da eliminare e quindi Elimina.
5. Nella pagina Elimina profilo Fargate, digita il nome del profilo e scegli Elimina.

## CLI AWS

Eliminare un profilo Fargate con AWS CLI

Utilizza il comando seguente per eliminare un profilo da un cluster. Sostituire ogni *`valore di esempio`* con i propri valori.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-
cluster
```

# Comprendere i dettagli di configurazione dei pod di Fargate

Questa sezione descrive alcuni dei dettagli di configurazione dei pod unici per l'esecuzione dei pod Kubernetes su AWS Fargate.

## Memoria e CPU del pod

Kubernetes ti consente di definire le richieste, una quantità minima di risorse vCPU e di memoria allocate a ciascun container in un pod. I pod sono programmati da Kubernetes per garantire che almeno le risorse richieste per ciascun pod siano disponibili nella risorsa di calcolo. Per ulteriori informazioni, consulta Gestione risorse di calcolo per container nella documentazione su Kubernetes.

> **Note**
>
> Poiché Amazon EKS Fargate esegue solo un pod per nodo, lo scenario di espulsione dei pod in caso di meno risorse non si verifica. Tutti i pod Amazon EKS Fargate funzionano con priorità garantita, quindi la CPU e la memoria richieste devono essere uguali al limite per tutti i container. Per ulteriori informazioni, consulta Gestire la qualità di servizio per i pod nella documentazione di Kubernetes.

Quando i pod sono programmati su Fargate, le prenotazioni di vCPU e memoria all'interno della specifica del pod determinano la quantità di CPU e memoria per effettuare il provisioning per il pod.

- La richiesta massima di qualsiasi container Init viene utilizzata per determinare i requisiti di memoria e vCPU della richiesta Init.
- Le richieste per tutti i container con esecuzione prolungata vengono aggiunte per determinare i requisiti di memoria e vCPU della richiesta con esecuzione prolungata.
- Il valore maggiore tra i due valori sopra indicati viene scelto per la richiesta di vCPU e memoria da utilizzare per il pod.
- Fargate aggiunge 256 MB alla prenotazione di memoria di ogni pod per i componenti Kubernetes richiesti (`kubelet`, `kube-proxy` e `containerd`).

Fargate arrotonda alla configurazione di calcolo mostrata di seguito che corrisponde più strettamente alla somma delle richieste di vCPU e di memoria, al fine di garantire che i pod abbiano sempre le risorse necessarie per la loro esecuzione.

Se non specifichi una combinazione di vCPU e memoria, viene utilizzata la combinazione più piccola disponibile (.25 vCPU e 0,5 GB di memoria).

La tabella seguente mostra le combinazioni di vCPU e memoria disponibili per i pod in esecuzione su Fargate.

| Valore vCPU | Valore memoria |
|---|---|
| .25 vCPU | 0,5 GB, 1 GB, 2 GB |
| .5 vCPU | 1 GB, 2 GB, 3 GB, 4 GB |
| 1 vCPU | 2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB |
| 2 vCPU | Tra 4 GB e 16 GB in incrementi di 1 GB |
| 4 vCPU | Tra 8 GB e 30 GB in incrementi di 1 GB |
| 8 vCPU | Tra 16 GB e 60 GB in incrementi di 4 GB |
| 16 vCPU | Tra 32 GB e 120 GB in incrementi di 8 GB |

La memoria aggiuntiva riservata ai componenti Kubernetes può causare un'attività Fargate con più vCPUs di quelle richieste per il provisioning. Ad esempio, una richiesta per 1 vCPU e 8 GB di memoria avrà 256 MB aggiunti alla richiesta di memoria e fornirà un'attività Fargate di 2 vCPU e 9 GB di memoria, poiché non è disponibile alcuna attività con 1 vCPU e 9 GB di memoria.

Non vi è alcuna correlazione tra la dimensione del pod in esecuzione su Fargate e quella del nodo riportata da Kubernetes con `kubectl get nodes`. La dimensione del nodo riportata è spesso maggiore della capacità del pod. Puoi verificare la capacità del pod utilizzando il comando seguente. Sostituisci *`default`* con il namespace del pod e *`pod-name`* con il nome dello stesso.

```
kubectl describe pod --namespace default pod-name
```

Di seguito viene riportato un output di esempio:

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
```

```
[...]
```

L'annotazione `CapacityProvisioned` rappresenta la capacità del pod applicata e determina il costo del pod in esecuzione su Fargate. Per informazioni sui prezzi di queste configurazioni di calcolo, vedi Prezzi Fargate AWS.

## Archiviazione in Fargate

Un pod in esecuzione su Fargate monta automaticamente un file system Amazon EFS senza bisogno della procedura di installazione manuale del driver. Non puoi utilizzare il provisioning dinamico dei volumi persistenti con nodi Fargate, ma puoi usare quello statico. Per ulteriori informazioni, consulta Driver CSI per Amazon EFS su GitHub.

Una volta fornito, ciascun pod in esecuzione su Fargate riceve uno storage temporaneo predefinito di 20 GiB. Questo tipo di archiviazione viene eliminato dopo che un pod si ferma. Per i nuovi pod avviati su Fargate, la crittografia del volume di archiviazione temporanea è abilitata per impostazione predefinita. L'archiviazione del pod effimero viene crittografata con un algoritmo di crittografia AES-256 utilizzando chiavi gestite da Fargate AWS.

> **Note**
>
> Lo storage utilizzabile predefinito per i pod Amazon EKS che funzionano su Fargate sono inferiori a 20 GiB. Questo perché parte dello spazio viene utilizzato da `kubelet` e altri moduli Kubernetes che vengono caricati all'interno del pod.

Puoi aumentare la quantità totale di storage temporaneo fino a un massimo di 175 GiB. Per configurare la dimensione con Kubernetes, specifica le richieste della risorsa `ephemeral-storage` per ogni container in un pod. Quando Kubernetes pianifica i pod, garantisce che la somma delle richieste di risorse per ciascun pod sia inferiore alla capacità dell'attività Fargate. Per ulteriori informazioni, consulta Gestione delle risorse per pod e container nella documentazione di Kubernetes.

Amazon EKS Fargate fornisce un maggiore spazio di archiviazione temporaneo di quello richiesto ai fini dell'utilizzo del sistema. Ad esempio, una richiesta di 100 GiB fornirà un'attività Fargate con 115 GiB di storage temporaneo.

# Imposta azioni per gli eventi relativi all'applicazione di patch del sistema operativo AWS Fargate

Amazon EKS applica periodicamente delle patch al sistema operativo per i nodi AWS Fargate al fine di garantirne la sicurezza. Come parte del processo di applicazione delle patch, ricicliamo i nodi per installare le patch del sistema operativo. Gli aggiornamenti vengono effettuati in modo da avere il minore impatto possibile sui servizi. Tuttavia, se i pod non vengono espulsi con successo, a volte è necessario eliminarli. Di seguito sono riportate le azioni che è possibile intraprendere per ridurre al minimo le potenziali interruzioni:

- Imposta i budget di interruzione dei pod (PDB) appropriati per limitare il numero di pod inattivi.
- Crea regole di Amazon EventBridge per gestire le espulsioni non riuscite prima che i pod vengano eliminati.
- Riavvia manualmente i pod interessati prima della data di espulsione indicata nella notifica ricevuta.
- Crea una configurazione di notifica in Notifiche utente di AWS.

Amazon EKS lavora a stretto contatto con la community di Kubernetes per rendere disponibili le correzioni dei bug e le patch di sicurezza il più rapidamente possibile. L'avvio di tutti i pod Fargate avviene con la versione più recente della patch di Kubernetes, disponibile da Amazon EKS per la versione di Kubernetes del cluster. Se disponi di un pod con una patch di una versione precedente, Amazon EKS potrebbe riciclarlo per aggiornarlo alla versione più recente. Ciò consente di applicare ai pod gli aggiornamenti per la sicurezza più recenti, in modo da ridurre i rischi nel caso in cui si verifichi una problematica di tipo Common Vulnerabilities and Exposures (CVE).

Quando il sistema operativo AWS Fargate viene aggiornato, Amazon EKS ti invia una notifica che include le risorse interessate e la data delle prossime espulsioni dei pod. Se la data di espulsione fornita è scomoda, hai la possibilità di riavviare manualmente i pod interessati prima della data di espulsione indicata nella notifica. Tutti i pod creati prima della data di ricezione della notifica sono soggetti a espulsione. Per ulteriori istruzioni su come riavviare manualmente i pod, consulta Kubernetes Documentation.

Per limitare il numero di pod inattivi durante il riciclo degli stessi, puoi impostare i budget di interruzione dei pod (PDB). Con questa opzione è possibile specificare la disponibilità minima in base ai requisiti di ciascuna applicazione e consentire, allo stesso tempo, l'esecuzione di aggiornamenti. La disponibilità minima del PDB deve essere inferiore al 100%. Per ulteriori informazioni, consulta Specifying a Disruption Budget for your Application nella documentazione di Kubernetes.

Amazon EKS utilizza l'API di espulsione per interrompere in modo sicuro il pod rispettando i budget di interruzione dei pod impostati per l'applicazione. I pod vengono espulsi dalla zona di disponibilità per ridurre al minimo l'impatto. Se l'espulsione ha esito positivo, il nuovo pod riceve la patch più recente e non sono richieste ulteriori operazioni.

Se l'espulsione di un pod ha esito negativo, Amazon EKS invia al tuo account un evento con dettagli relativi al pod in cui si è verificato l'errore. È possibile intraprendere alcune azioni in merito al messaggio prima del periodo di interruzione programmato, che varia in base all'urgenza della patch. Quando è il momento, Amazon EKS proverà a espellere nuovamente i pod. Questa volta, tuttavia, se l'espulsione ha esito negativo non viene inviato un nuovo evento Se l'espulsione continua a non riuscire, i pod esistenti verranno eliminati periodicamente in modo che i nuovi pod possano disporre della patch più recente.

L'esempio seguente mostra un evento ricevuto in caso di espulsione del pod che non riesce. Il messaggio contiene dettagli sul cluster, il nome del pod, il namespace del pod, il profilo Fargate e il periodo di interruzione pianificato.

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

L'associazione di molteplici PDB a un singolo pod può causare un evento di errore di espulsione. Questo evento restituisce il messaggio di errore seguente.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

È possibile creare un'azione desiderata in base a questo evento. Ad esempio, puoi modificare il budget di interruzione dei pod (PDB) per controllare il modo in cui i pod vengono espulsi. Più specificamente, supponi di iniziare con un PDB che specifica la percentuale di destinazione dei pod disponibili. Prima che i pod vengano terminati in modo forzato durante un aggiornamento, puoi regolare il PDB su una percentuale di pod differente. Per ricevere questo evento, devi creare una regola di Amazon EventBridge nell'account AWS e nella Regione AWS a cui appartiene il cluster. La regola deve utilizzare il modello personalizzato seguente. Per ulteriori informazioni, consulta la sezione Creazione di regole Amazon EventBridge che reagiscono agli eventi nella Guida per l'utente di Amazon EventBridge.

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

L'evento può essere configurato per acquisire un obiettivo adeguato impostato dall'utente. Per un elenco completo delle destinazioni disponibili, consulta Destinazioni di Amazon EventBridge nella Guida per l'utente di Amazon EventBridge. Inoltre puoi creare una configurazione di notifica in Notifiche utente AWS. Quando usi Console di gestione AWS per creare la notifica, sotto la voce Regole degli eventi scegli Elastic Kubernetes Service (EKS) per il Nome di servizio AWS e Arresto programmato di EKS Fargate pod per il Tipo di evento. Per ulteriori informazioni, consulta Guida introduttiva alle Notifiche utente AWS nella Guida per l'utente delle Notifiche utente AWS.

Consulta Domande frequenti: Avviso di espulsione di un Fargate Pod in AWS re:Post per le domande frequenti riguardo alle espulsioni dei pod EKS.

## Raccogli le AWS metriche dell'app Fargate e dell'utilizzo

Puoi raccogliere metriche di sistema e metriche di CloudWatch utilizzo per AWS Fargate.

### Parametri di applicazione

Per le applicazioni in esecuzione su Amazon EKS e AWS Fargate, puoi utilizzare AWS Distro for OpenTelemetry (ADOT). ADOT ti consente di raccogliere i parametri di sistema e inviarli ai dashboard di Container Insights. CloudWatch Per iniziare a usare ADOT per le applicazioni

in esecuzione su Fargate, consultate CloudWatch Using Container Insights AWS with Distro OpenTelemetry for nella documentazione ADOT.

## Parametri di utilizzo

Puoi utilizzare le metriche di CloudWatch utilizzo per fornire visibilità sull'utilizzo delle risorse da parte del tuo account. Utilizza queste metriche per visualizzare l'utilizzo corrente del servizio su CloudWatch grafici e dashboard.

AWS Le metriche di utilizzo di Fargate corrispondono alle AWS quote di servizio. È possibile configurare gli allarmi che avvisano quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni sulle quote di servizio per Fargate, consulta the section called "Service Quotas".

AWS Fargate pubblica le seguenti metriche nel namespace. `AWS/Usage`

| Metrica | Description |
| --- | --- |
| `ResourceCount` | Il numero totale delle risorse specificate in esecuzione nell'account. La risorsa è definita dalle dimensioni associate attraverso il parametro. |

Le seguenti dimensioni vengono utilizzate per perfezionare le metriche di utilizzo pubblicate da AWS Fargate.

| Dimensione | Description |
| --- | --- |
| `Service` | Il nome del AWS servizio che contiene la risorsa. Per le metriche di utilizzo di AWS Fargate, il valore per questa dimensione è. `Fargate` |
| `Type` | Il tipo di entità che viene segnalato. Attualmente, l'unico valore valido per le metriche di utilizzo di AWS Fargate è. `Resource` |
| `Resource` | Il tipo di risorsa in esecuzione. |

| Dimensione | Description |
| --- | --- |
| | Attualmente, AWS Fargate restituisce informazioni sull'utilizzo di Fargate On-Demand. Il valore della risorsa per l'utilizzo on demand di Fargate è `OnDemand`.<br><br>[NOTA] ====<br><br>L'utilizzo di Fargate On-Demand combina i pod Amazon EKS utilizzando Fargate, le attività Amazon ECS che utilizzano il tipo di avvio di Fargate e le attività Amazon ECS che usano il provider di capacità `FARGATE`.<br><br>==== |
| `Class` | La classe della risorsa monitorata. Attualmente, AWS Fargate non utilizza la dimensione classe. |

Creazione di un CloudWatch allarme per monitorare le metriche di utilizzo delle risorse di Fargate

AWS Fargate fornisce metriche di CloudWatch utilizzo che corrispondono alle quote di AWS servizio per l'utilizzo delle risorse Fargate On-Demand. Nella console Service Quotas (Quote di Servizio) è possibile visualizzare l'utilizzo in un grafico. È inoltre possibile configurare gli allarmi che avvisano quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni, consulta the section called "Raccolta dei parametri".

Utilizzate i seguenti passaggi per creare un CloudWatch allarme basato sulle metriche di utilizzo delle risorse di Fargate.

1. Apri la console Service Quotas all'indirizzo. https://console.aws.amazon.com/servicequotas/
2. Nel riquadro di navigazione a sinistra, scegli  AWS servizi.
3. Dall'elenco dei  AWS servizi, cerca e seleziona  AWS Fargate.
4. Nell'elenco Service Quotas, seleziona la quota di utilizzo Fargate per cui desideri creare un allarme.
5. Nella sezione CloudWatch Allarmi Amazon, scegli Crea.

6. Per Soglia di allarme, scegli la percentuale del valore della quota applicata che desideri impostare come valore per l'allarme.
7. Per Nome allarme, immetti un nome per l'allarme e quindi scegli Crea.

## Start (Avvio) AWS Registrazione Fargate per il tuo cluster

Amazon EKS su Fargate offre un router di registro integrato basato su Fluent Bit. Ciò significa che non viene eseguito esplicitamente un container Fluent Bit come sidecar, ma è Amazon a eseguirlo. Tutto quello che devi fare è configurare il router di log. La configurazione avviene attraverso un `ConfigMap` dedicato che deve soddisfare i seguenti criteri:

- Deve essere denominato `aws-logging`
- Creato in un namespace dedicato denominato `aws-observability`
- Non può contenere più di 5.300 caratteri.

Una volta creato `ConfigMap`, Amazon EKS su Fargate lo rileva in automatico e lo utilizza per configurare il router di log. Fargate utilizza una versione di AWS for Fluent Bit, una distribuzione conforme agli standard upstream di Fluent Bit gestita da. AWS Per ulteriori informazioni, vedere Fluent Bit on. AWS GitHub

Il router di log consente di utilizzare l'ampia gamma di servizi di at AWS per l'analisi e l'archiviazione dei log. Puoi trasmettere i log da Fargate direttamente ad Amazon CloudWatch e Amazon Service. OpenSearch È possibile trasmettere i log anche a destinazioni come Amazon S3 e flusso di dati Amazon Kinesis, così come a strumenti partner tramite Amazon Data Firehose.

- Un profilo Fargate esistente che specifichi un namespace Kubernetes esistente in cui si implementano i pod Fargate. Per ulteriori informazioni, consulta the section called “Fase 3: creare un profilo Fargate per il cluster”.
- Un ruolo di esecuzione del pod Fargate esistente. Per ulteriori informazioni, consulta the section called “Fase 2: creare un ruolo di esecuzione del pod Fargate”.

## Configurazione del router di log

> **Important**
>
> Affinché i log vengano pubblicati correttamente, deve essere possibile accedere alla rete dal VPC in cui si trova il cluster alla destinazione dei log. Ciò riguarda principalmente gli utenti che personalizzano le regole di uscita per il VPC. Per un esempio di utilizzo CloudWatch, consulta Using CloudWatch Logs with interface VPC endpoints nella Amazon CloudWatch Logs User Guide.

Nei passaggi che seguono, sostituire *example value* con i propri valori.

1. Creare un namespace Kubernetes dedicato denominato `aws-observability`.
   a. Salva nel tuo computer i seguenti contenuti in un file denominato `aws-observability-namespace.yaml`. Il valore per `name` deve essere `aws-observability` e l'etichetta `aws-observability: enabled` è obbligatoria.

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
    aws-observability: enabled
```

   b. Crea il namespace.

```
kubectl apply -f aws-observability-namespace.yaml
```

2. Crea una `ConfigMap` con valore dei dati `Fluent Conf` per spedire i log del container verso una destinazione. Fluent Conf è Fluent Bit, un linguaggio di configurazione del processore di log veloce e leggero che viene utilizzato per instradare i log del container verso una destinazione di log selezionata. Per ulteriori informazioni, consultare File di configurazione nella documentazione di Fluent Bit.

> **Important**
>
> In una `Fluent Conf` tipica, le sezioni principali incluse sono `Service`, `Input`, `Filter` e `Output`. Tuttavia, il router di log di Fargate accetta solo:

- Le sezioni `Filter` e `Output`.
- Una sezione `Parser`.

Se fornisci altre sezioni, queste verranno rifiutate.

Il router di log Fargate gestisce le sezioni `Service` e `Input`. Ha la sezione `Input` seguente, che non può essere modificata e non è necessaria in `ConfigMap`. Tuttavia, è possibile ricavarne informazioni, ad esempio il limite del buffer di memoria e il tag applicato per i log.

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

Quando crei la `ConfigMap`, ricorda le seguenti regole utilizzate da Fargate per convalidare i campi:

- `[FILTER]`, `[OUTPUT]`, e `[PARSER]` dovrebbero essere specificati sotto ogni chiave corrispondente. Ad esempio: `[FILTER]` deve essere in `filters.conf`. È possibile avere uno o più `[FILTER]` in `filters.conf`. Anche le sezioni `[OUTPUT]` e `[PARSER]` dovrebbero essere specificate sotto le chiavi corrispondenti. Specificando più sezioni `[OUTPUT]`, è possibile instradare i registritha a destinazioni diverse contemporaneamente.
- Fargate convalida le chiavi richieste per ogni sezione. `Name` e `match` sono necessari per ogni `[FILTER]` e `[OUTPUT]`. `Name` e `format` sono necessari per ogni `[PARSER]`. Le chiavi non fanno distinzione tra maiuscole e minuscole.
- Variabili di ambiente come `${ENV_VAR}` non sono ammesse in `ConfigMap`.
- L'indentazione deve essere la stessa per la direttiva o per la coppia chiave-valore all'interno di ciascuna, e. `filters.conf` `output.conf` `parsers.conf` Key-value le coppie devono essere indentate più delle direttive.
- Fargate effettua la convalida in base ai seguenti filtri supportati: `grep`, `parser`, `record_modifier`, `rewrite_tag`, `throttle`, `nest`, `modify` e `kubernetes`.

- Fargate effettua la convalida in base al seguente output supportato: `es`, `firehose`, `kinesis_firehose`, `cloudwatch`, `cloudwatch_logs`, e `kinesis`.
- In `ConfigMap` deve essere fornito almeno un plugin `Output` supportato per abilitare la registrazione di log. `Filter` e `Parser` non sono necessari per abilitarla.

  È possibile anche eseguire Fluent Bit su Amazon EC2 utilizzando la configurazione desiderata per risolvere eventuali problemi derivanti dalla convalida. Crea il `ConfigMap` utilizzando uno degli esempi seguenti.

  > **Important**
  >
  > La registrazione di log Fargate di Amazon EKS non supporta la configurazione dinamica di `ConfigMap`. Eventuali modifiche a `ConfigMap` vengono applicate solo ai nuovi pod. Le modifiche non vengono applicate ai pod esistenti.

  Crea un `ConfigMap` utilizzando l'esempio per la destinazione di log desiderata.

  > **Note**
  >
  > Puoi anche utilizzare Flusso di dati Amazon Kinesis come destinazione dei log. Se utilizzi Flusso di dati Kinesis, assicurati che al ruolo di esecuzione del pod sia stata concessa l'autorizzazione `kinesis:PutRecords`. Per ulteriori informazioni, consulta la sezione Permissions relativa a flusso di dati Amazon Kinesis in Fluent Bit: Official Manual.

Example

CloudWatch

Sono disponibili due opzioni di output quando si utilizza: CloudWatch

- Un plug-in di output scritto in C
- Un plug-in di output scritto in Golang

L'esempio seguente mostra come utilizzare il `cloudwatch_logs` plug-in per inviare i log a CloudWatch.

a. Salva i contenuti seguenti in un file denominato `aws-logging-cloudwatch-configmap.yaml`. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster. I parametri in `[OUTPUT]` sono obbligatori.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

b. Applica il file manifesto al cluster.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

Amazon OpenSearch Service

Se desideri inviare log ad Amazon OpenSearch Service, puoi utilizzare es output, un plugin scritto in C. L'esempio seguente mostra come utilizzare il plug-in a cui inviare i log OpenSearch.

a. Salva i contenuti seguenti in un file denominato `aws-logging-opensearch-configmap.yaml`. Sostituisci ogni *`example value`* con i valori in tuo possesso.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

b. Applica il manifesto al cluster.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

Firehose

Sono disponibili due opzioni di output per l'invio di log a Firehose:

- kinesis_firehose: un plugin di output scritto in C.
- firehose: un plugin di output scritto in Golang.

Nell'esempio seguente viene illustrato come utilizzare il plugin `kinesis_firehose` per inviare log a Firehose.

a. Salva i contenuti seguenti in un file denominato `aws-logging-firehose-configmap.yaml`. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
     Name  kinesis_firehose
     Match *
     region region-code
     delivery_stream my-stream-firehose
```

b. Applica il file manifesto al cluster.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Configurare le autorizzazioni per il ruolo di esecuzione del pod Fargate per inviare log alla destinazione.

a. Eseguire il download della policy IAM per la destinazione sul computer.

Example

CloudWatch

Scarica la policy CloudWatch IAM sul tuo computer. Puoi anche visualizzare la politica su GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Scarica la policy OpenSearch IAM sul tuo computer. Puoi anche visualizzare la politica su GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

Assicurati che il controllo OpenSearch degli accessi di Dashboards sia configurato correttamente. Le OpenSearch dashboard `all_access role` in devono avere il ruolo di esecuzione di Fargate Pod e il ruolo IAM mappati. La stessa mappatura deve essere eseguita per il ruolo `security_manager`. È possibile aggiungere i mapping precedenti selezionando `Menu`, `Security`, `Roles`, quindi selezionare i rispettivi ruoli. Per ulteriori informazioni, consulta Come posso risolvere i problemi relativi ai CloudWatch log in modo che vengano trasmessi al mio dominio Amazon ES? .

Firehose

Eseguire il download della policy IAM Firehose sul computer Puoi anche visualizzare la politica su GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

b. Creare una policy IAM utilizzando il file della policy scaricato nella fase precedente.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

c. Collega la policy IAM al ruolo di esecuzione del pod specificato per il profilo Fargate con il seguente comando. Sostituisci *111122223333* con l'ID del tuo account. Sostituiscilo *AmazonEKSFargatePodExecutionRole* con il tuo ruolo di esecuzione del Pod (per ulteriori informazioni, consultathe section called "Fase 2: creare un ruolo di esecuzione del pod Fargate").

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Supporto filtri Kubernetes

Il filtro Fluent Bit Kubernetes consente di aggiungere metadati Kubernetes ai file di log. Per ulteriori informazioni sul filtro, consultare Kubernetes nella documentazione di Fluent Bit. È possibile applicare un filtro utilizzando l'endpoint del server API.

```
filters.conf: |
    [FILTER]
        Name             kubernetes
        Match            kube.*
        Merge_Log           On
        Buffer_Size         0
        Kube_Meta_Cache_TTL 300s
```

**Important**

- `Kube_URL`, `Kube_CA_File`, `Kube_Token_Command` e `Kube_Token_File` sono parametri di configurazione di proprietà del servizio e non devono essere specificati. Amazon EKS Fargate popola questi valori.
- `Kube_Meta_Cache_TTL` è il momento in cui Fluent Bit attende il momento in cui comunica al server API i metadati più recenti. Se `Kube_Meta_Cache_TTL` non è specificato, Amazon EKS Fargate aggiunge un valore predefinito di 30 minuti per ridurre il carico sul server API.

Per inviare log di processo Fluent Bit all'account

Facoltativamente, puoi inviare i log dei processi di Fluent Bit ad Amazon CloudWatch utilizzando quanto segue. `ConfigMap` La spedizione dei log di processo di Fluent Bit CloudWatch richiede costi aggiuntivi di inserimento e archiviazione dei log. Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
```

```
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

I log si trovano CloudWatch nella stessa AWS regione del cluster. Il nome del gruppo di log è *my-cluster*`-fluent-bit-logs` e il nome del flusso di log Fluent Bit è `fluent-bit-`*podname*`-`*pod-namespace* .

> **Note**
>
> - I log dei processi vengono spediti solo quando il processo Fluent Bit viene avviato correttamente. Se si verifica un errore durante l'avvio di Fluent Bit, i log di processo vengono mancati. È possibile inviare i log dei processi solo a CloudWatch.
> - Per eseguire il debug della spedizione dei log di processo sull'account, è possibile applicare la `ConfigMap` precedente per ottenere i log di processo. Il mancato avvio di Fluent Bit solitamente è dovuto al fatto che la `ConfigMap` non viene analizzata o accettata da Fluent Bit durante l'avvio.

Per interrompere l'invio dei log di processo Fluent Bit

La spedizione dei log di processo di Fluent Bit CloudWatch richiede costi aggiuntivi di inserimento e archiviazione dei log. Per escludere i registri dei processii in una configurazione `ConfigMap` esistente, effettua le seguenti operazioni.

1. Individua il gruppo di CloudWatch log creato automaticamente per i log di processo Fluent Bit del tuo cluster Amazon EKS dopo aver abilitato la registrazione di Fargate. Segue il formato *my-cluster*`-fluent-bit-logs`.

2. Elimina i flussi di CloudWatch log esistenti creati per i log di processo di ciascun Pod nel gruppo di log. CloudWatch
3. Modifica la `ConfigMap` e imposta `flb_log_cw: "false"`.
4. Riavviare tutti i pod esistenti nel cluster.

## Applicazione di prova

1. Implementare un pod di esempio.
   a. Salva nel tuo computer i seguenti contenuti in un file denominato `sample-app.yaml`.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

   b. Applica il file manifesto al cluster.

```
kubectl apply -f sample-app.yaml
```

2. Visualizza i log NGINX utilizzando la/le destinazione/i configurata/e nel `ConfigMap`.

### Considerazioni sulle dimensioni

Si consiglia di pianificare fino a 50 MB di memoria per il router di log. Se prevedi che l'applicazione generi registri a velocità di trasmissione effettiva molto elevata, dovresti pianificarla fino a 100 MB.

### Risoluzione dei problemi

Per confermare se la funzionalità di registrazione di log è abilitata o disabilitata per qualche motivo, ad esempio per una `ConfigMap` non valida, e perché questa non è valida, controllare gli eventi pod con `kubectl describe pod` *`pod-name`* . L'output potrebbe includere eventi pod che chiariscono se la registrazione di log è abilitata o meno, ad esempio l'output di esempio seguente.

```
[...]
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
[...]
Events:
  Type     Reason           Age        From
               Message
  ----     ------           ----       ----
               -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
                Disabled logging because aws-logging configmap was not found. configmap
 "aws-logging" not found
```

Gli eventi di pod sono effimeri, con un periodo di tempo che dipende dalle impostazioni. È inoltre possibile visualizzare le annotazioni di un pod tramite `kubectl describe pod` *`pod-name`* . Nell'annotazione del pod, è possibile sapere se, e per quale motivo, la funzionalità di registrazione di log è abilitata o disabilitata.

## Scelta di una tipologia di istanza di nodo Amazon EC2 ottimale

Amazon EC2 offre un'ampia gamma di tipi di istanze per i nodi worker. Ogni tipo di istanza mette a disposizione diverse capacità di calcolo, memoria, archiviazione e rete ed è raggruppata in una famiglia di istanze in base a tali capacità. Per un elenco, consultare Available instance types nella Guida per l'utente di Amazon EC2 per le istanze Linux. Amazon EKS rilascia diverse varianti delle AMI Amazon EC2 per abilitare il supporto. Per assicurarti che il tipo di istanza selezionato sia compatibile con Amazon EKS, prendi in considerazione gli elementi seguenti:

- Al momento, nessuna AMI Amazon EKS supporta la famiglia `mac`.

- Le AMI ARM e Amazon EKS non accelerate non supportano le famiglie `g3`, `g4`, `inf` e `p`.
- Le AMI Amazon EKS accelerate non supportano le famiglie `a`, `c`, `hpc`, `m` e `t`.
- Ad Arm-based esempio, Amazon Linux 2023 (AL2023) supporta solo tipi di istanza che utilizzano processori Graviton2 o successivi. AL2023 non supporta le istanze `A1`.

Quando scegli i tipi di istanza supportati da Amazon EKS, considera le caratteristiche seguenti per ogni tipo.

Numero di istanze in un gruppo di nodi

In generale, istanze in numero ridotto e di grandi dimensioni sono più convenienti, specialmente disponi di molti DaemonSet. Ogni istanza richiede chiamate API al server API, per cui maggiore è il numero di istanze di cui si dispone, maggiore è il carico sul server API.

Sistema operativo

Esamina i tipi di istanza supportati per Linux, Windows e Bottlerocket. Prima di creare istanze Windows, consultare Deploy Windows nodes on EKS clusters.

Architettura hardware

Hai bisogno di x86 o ARM? Prima di implementare istanze ARM, consultare Amazon EKS optimized Arm Amazon Linux AMIs. Hai bisogno di istanze costruite sul sistema Nitro (Linux o Windows) o che hanno una elaborazione Accelerata? Se hai bisogno di funzionalità accelerate, puoi usare Linux solo con Amazon EKS.

Numero massimo di pod

Dal momento che a ogni Pod viene assegnato il proprio indirizzo IP, il numero di indirizzi IP supportati da un tipo di istanza è un fattore importante per determinare il numero di pod che possono essere eseguiti sull'istanza. Per capire come viene determinato il numero massimo di Pod per un tipo di istanza, consulta. the section called "Come viene determinato MaxPods"

AWS I tipi di istanza di Nitro System supportano opzionalmente un numero significativamente maggiore di indirizzi IP rispetto ai tipi di istanze non Nitro System. Tuttavia, non tutti gli indirizzi IP assegnati ad un'istanza sono disponibili per i Pod. Per assegnare un numero significativamente maggiore di indirizzi IP alle istanze, è necessario che la versione `1.9.0` o successiva del componente aggiuntivo CNI di Amazon VPC sia installata nel cluster e che sia configurata in modo appropriato. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP". Per assegnare il maggior numero di indirizzi IP alle istanze, è necessario che la versione

`1.10.1` o successiva del componente aggiuntivo CNI di Amazon VPC sia installata nel cluster e che si implementi il cluster con la famiglia `IPv6`.

Famiglia IP

Per la creazione di un cluster tramite la famiglia `IPv4`, puoi scegliere qualsiasi tipo di istanza supportata, il che consente al cluster di assegnare indirizzi `IPv4` privati ai Pod e servizi. Se tuttavia desideri impiegare la famiglia `IPv6` per il cluster, utilizza i tipi di istanza AWS Nitro System o bare metal. Per le istanze Windows è supportato solo `IPv4`. Il cluster deve eseguire la versione `1.10.1` o successiva del componente aggiuntivo CNI di Amazon VPC. Per ulteriori informazioni sull'utilizzo di `IPv6`, consultare the section called "IPv6".

Versione del componente aggiuntivo CNI di Amazon VPC in uso

La versione più recente del plug-in CNI di Amazon VPC per Kubernetes supporta i seguenti tipi di istanza. Per sfruttare i tipi più recenti di istanza supportati, potrebbe essere necessario aggiornare la versione del componente aggiuntivo CNI di Amazon VPC. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC". L'ultima versione supporta le funzionalità più recenti per l'utilizzo con Amazon EKS. Le versioni precedenti non supportano tutte le funzionalità. Puoi visualizzare le funzionalità supportate dalle diverse versioni in Changelog su GitHub.

La Regione AWS in cui stai creando i nodi

Non tutti i tipi di istanza sono disponibili in tutte le AWS regioni.

Se stai usando gruppi di sicurezza per i Pod

Se stai usando gruppi di sicurezza per i Pod, sono supportati solo tipi specifici di istanza. Per ulteriori informazioni, consulta the section called "Gruppi di sicurezza per pod".

## Come viene determinato MaxPods

Il `maxPods` valore finale applicato a un nodo dipende da diversi componenti che interagiscono in uno specifico ordine di precedenza. La comprensione di questo ordine consente di evitare comportamenti imprevisti durante la personalizzazione`maxPods`.

Ordine di precedenza (dal più alto al più basso):

1. Applicazione gestita dei gruppi di nodi: quando si utilizza un gruppo di nodi gestiti senza un'AMI personalizzata, Amazon EKS impone `maxPods` un limite ai dati utente del nodo. Per le istanze con meno di 30 vCPU, il limite è. `110` Per le istanze con più di 30 vCPU, il limite è. `250` Questo valore ha la precedenza su qualsiasi altra `maxPods` configurazione, inclusa. `maxPodsExpression`

2. **`maxPods`**configurazione kubelet — Se si imposta `maxPods` direttamente nella configurazione kubelet (ad esempio, tramite un modello di avvio con un'AMI personalizzata), questo valore ha la precedenza su. `maxPodsExpression`
3. nodeadm **`maxPodsExpression`** — Se lo usi `maxPodsExpression` nel tuo`NodeConfig`, nodeadm valuta l'espressione da calcolare. `maxPods` Questo è efficace solo quando il valore non è già impostato da una fonte con precedenza più alta.
4. ENI-based Calcolo predefinito: se non viene impostato nessun altro valore, l'AMI calcola in `maxPods` base al numero di interfacce di rete elastiche e di indirizzi IP supportati dal tipo di istanza. È equivalente alla formula. `(number of ENIs × (IPs per ENI − 1)) + 2` Gli `+ 2` account di Amazon VPC CNI sono `kube-proxy` attivi su tutti i nodi e non utilizzano un indirizzo IP Pod.

> **Important**
>
> Se utilizzi un gruppo di nodi gestiti e lo `maxPodsExpression` impostiNodeConfig, l'applicazione del gruppo di nodi gestiti sovrascrive la tua espressione. Per utilizzare un `maxPods` valore personalizzato con i gruppi di nodi gestiti, devi specificare un'AMI personalizzata nel modello di avvio e `maxPods` impostarla direttamente. Per ulteriori informazioni, consulta the section called “Modelli di avvio”.

Gruppi di nodi gestiti e nodi autogestiti

Con i gruppi di nodi gestiti (senza un'AMI personalizzata), Amazon EKS inserisce il `maxPods` valore nei dati utente bootstrap del nodo. Ciò significa che:

- Il `maxPods` valore è sempre limitato `110` o `250` dipende dalla dimensione dell'istanza.
- Qualsiasi `maxPodsExpression` configurazione viene sovrascritta da questo valore iniettato.
- Per utilizzare un `maxPods` valore diverso, specifica un'AMI personalizzata nel modello di avvio e passala `--use-max-pods false --kubelet-extra-args '--max-pods=my-value'` allo script. `bootstrap.sh` Per alcuni esempi, consulta the section called “Modelli di avvio”.

Con i nodi autogestiti, hai il pieno controllo del processo di bootstrap. Puoi utilizzare `maxPodsExpression` nel tuo `NodeConfig` o passare `--max-pods` direttamente a. `bootstrap.sh`

## Considerazioni per la modalità automatica di EKS

modalità automatica di EKS limita il numero di pod sui nodi al valore inferiore tra:

- Tappo rigido da 110 pod
- `number of ENIs × (IPs per ENI - 1)`

# Creazione di nodi con immagini ottimizzate preconfigurate

Puoi implementare i nodi con versioni preconfigurate delle Amazon Machine Image (AMI) ottimizzate per Amazon EKS o con AMI personalizzate quando utilizzi i gruppi di nodi gestiti o i nodi autogestiti. Se stai eseguendo nodi ibridi, consulta the section called "Preparazione del sistema operativo". Per ulteriori informazioni sui vari tipi di AMI ottimizzate per Amazon EKS, consulta il relativo argomento di seguito. Per istruzioni su come creare un'AMI personalizzata, consulta the section called "Crea un'AMI EKS-optimized Amazon Linux personalizzata".

Con Amazon EKS Auto Mode, EKS gestisce l'istanza EC2, inclusa la selezione e l'aggiornamento dell'AMI.

Argomenti

- Guida alle funzionalità di transizione delle & AL2-Accelerated AMI EKS AL2
- Crea nodi con Amazon Linux ottimizzato AMIs
- Creazione di nodi con AMI Bottlerocket ottimizzate
- Creare nodi con AMI ottimizzate per Ubuntu Linux
- Crea nodi con Windows ottimizzato AMIs

## Guida alle funzionalità di transizione delle & AL2-Accelerated AMI EKS AL2

> **Warning**
>
> Amazon EKS ha smesso di pubblicare AMI EKS-optimized Amazon Linux 2 (AL2) il 26 novembre 2025. Le AMI basate su AL2023 e Bottlerocket per Amazon EKS sono disponibili per tutte le versioni di Kubernetes supportate, inclusa la 1.33 e successive.

AWS terminerà il supporto per EKS AL2-optimized e AL2-accelerated AMI a partire dal 26 novembre 2025. Sebbene sia comunque possibile continuare a utilizzare le AMI EKS AL2 dopo la data di fine del supporto (26 novembre 2025), dopo tale data EKS non rilascerà più nuove versioni o aggiornamenti di Kubernetes alle AMI AL2, comprese versioni secondarie, patch e correzioni di bug. Consigliamo di eseguire l'aggiornamento alle AMI Amazon Linux 2023 (AL2023) o Bottlerocket:

- AL2023 consente un approccio sicuro per impostazione predefinita con politiche di sicurezza preconfigurate, SELinux in modalità permissiva, IMDSv2-only modalità abilitata per impostazione predefinita, tempi di avvio ottimizzati e una migliore gestione dei pacchetti per una maggiore sicurezza e prestazioni, adatto per infrastrutture che richiedono personalizzazioni significative come accesso diretto o modifiche estese ai nodi. OS-level Per ulteriori informazioni, consulta le domande frequenti su AL2023 o visualizza la nostra guida dettagliata sulla migrazione all'indirizzo. the section called "Aggiornamento ad AL2023"
- Bottlerocket offre una sicurezza avanzata, tempi di avvio più rapidi e una superficie di attacco minore per una maggiore efficienza grazie al design progettato appositamente e ottimizzato per i container, adatto ad approcci nativi per i container con personalizzazioni minime dei nodi. Per saperne di più, consulta le domande frequenti su Bottlerocket o visualizza la nostra guida dettagliata sulla migrazione all'indirizzo. the section called "Portabottiglie"

In alternativa, puoi farlo the section called "Crea un'AMI EKS-optimized Amazon Linux personalizzata" fino alla data EOS (26 novembre 2025). Inoltre, puoi creare un'AMI personalizzata con un'istanza di base Amazon Linux 2 fino alla data EOS di Amazon Linux 2 (30 giugno 2026).

## Domande frequenti su migrazione e supporto

Come si esegue la migrazione da AL2 a un'AMI AL2023?

Consigliamo di creare e implementare un piano di migrazione che includa test approfonditi del carico di lavoro delle applicazioni e procedure di rollback documentate, quindi seguire le istruzioni dettagliate contenute nella pagina Upgrade from Amazon Linux 2 to Amazon Linux 2023 della documentazione ufficiale di EKS.

# possibile creare un'AMI AL2 personalizzata dopo la data di fine del supporto di EKS per le AMI AL2 ottimizzate per EKS?

Sebbene ti consigliamo di passare a AMI ottimizzate EKS ufficialmente supportate e pubblicate per AL2023 o Bottlerocket, puoi creare AL2-optimized EKS AL2-accelerated e AMI personalizzati fino alla data EOS dell'AMI AL2 (26 novembre 2025). In alternativa, è possibile creare un'AMI

personalizzata con un'istanza base di Amazon Linux 2 fino alla data di fine del supporto di Amazon Linux 2 (30 giugno 2026). Per istruzioni dettagliate su come creare un EKS e un' AL2-optimized AMI personalizzati, consulta Creare un' AL2-accelerated AMI Amazon Linux personalizzata nella documentazione ufficiale di EKS.

La policy di supporto delle versioni Kubernetes di EKS si applica alle distribuzioni Amazon Linux?

No. La data EOS per EKS AL2-optimized e AL2-accelerated AMI è indipendente dalle tempistiche di supporto standard ed estese per le versioni Kubernetes di EKS. È necessario eseguire la migrazione ad AL2023 o Bottlerocket anche se si usufruisce del supporto esteso di EKS.

In che modo il passaggio da cgroupv1 a cgroupv2 influisce sulla migrazione?

La community di Kubernetes ha trasferito il supporto di `cgroupv1` (utilizzato da AL2) in modalità manutenzione, il che significa che non verranno aggiunte nuove funzionalità e verranno forniti solo gli aggiornamenti di sicurezza critici e le correzioni di bug più importanti. Per adottare `cgroupv2` in Kubernetes, è necessario garantire la compatibilità tra sistema operativo, kernel, runtime dei container e componenti Kubernetes. Ciò richiede una distribuzione Linux con abilitazione di `cgroupv2` per impostazione predefinita, come AL2023, Bottlerocket, Red Hat Enterprise Linux (RHEL) 9+, Ubuntu 22.04+ o Debian 11+. Tali distribuzioni vengono fornite con versioni del kernel pari o superiori alla 5.8, che è il requisito minimo per il supporto di `cgroupv2` in Kubernetes. Per ulteriori informazioni, consultare la pagina About cgroup v2.

Cosa fare nel caso in cui sia necessario Neuron nell'AMI AL2 personalizzata?

Non è possibile eseguire le applicazioni complete in modo nativo su un'AMI. Neuron-powered AL2-based Per utilizzare AWS Neuron su un'AMI AL2, devi containerizzare le tue applicazioni utilizzando Neuron-supported un contenitore con una distribuzione Linux non AL2 (ad esempio, Ubuntu 22.04, Amazon Linux 2023, ecc.) e quindi distribuire quei contenitori su un' AL2-based AMI su cui è installato il driver Neuron (). `aws-neuronx-dkms`

# necessario passare a un'istanza base Amazon Linux 2 bare dopo la data di fine del supporto per le AMI EKS AL2 (26 novembre 2025)?

Il passaggio a un'istanza base di Amazon Linux 2 non include le ottimizzazioni specifiche, le configurazioni di runtime dei container e le personalizzazioni fornite dagli EKS e dalle AMI ufficiali. AL2-optimized AL2-accelerated Invece, se devi continuare a utilizzare una AL2-based soluzione, ti consigliamo di creare un'AMI personalizzata utilizzando le ricette AMI EKS su the section called "Crea un'AMI EKS-optimized Amazon Linux personalizzata" o Amazon EKS AMI Build Specification.

Ciò garantisce la compatibilità con i carichi di lavoro esistenti e include aggiornamenti del kernel AL2 fino alla data di fine del supporto di Amazon Linux 2 (30 giugno 2026).

Quando si crea un'AMI AL2 personalizzata utilizzando l' GitHub archivio EKS AMI dopo la data EKS AL2 AMI EOS (26 novembre 2025), quale supporto è disponibile per i pacchetti di repository come amzn2-core e amzn2extra-docker?

La ricetta di AMI EKS disponibile alla pagina Amazon EKS AMI Build Specification estrae pacchetti tramite YUM da software Amazon Linux 2 standard, come amzn2-core e amzn2extra-docker. Dopo la data di fine del supporto per le AMI EKS AL2 (26 novembre 2025), questo software continuerà a essere supportato fino alla data di fine del supporto di Amazon Linux 2 (30 giugno 2026). Tenere presente che durante questo periodo il supporto è limitato agli aggiornamenti del kernel, il che significa che sarà necessario gestire e applicare manualmente altri aggiornamenti dei pacchetti, patch di sicurezza e qualsiasi dipendenza non relativa al kernel per mantenere la sicurezza e la compatibilità.

Perché in applicazioni Java che utilizzano versioni precedenti di JDK8 su Amazon EKS con AL2023 potrebbero verificarsi eccezioni di memoria insufficiente (OOM) e riavvii dei pod? Come è possibile risolvere questo problema?

Quando vengono eseguite su nodi Amazon EKS con AL2023, le applicazioni Java basate su versioni di JDK 8 precedenti a `jdk8u372` possono causare eccezioni OOM e riavvii dei pod perché la JVM non è compatibile con `cgroupv2`. Questo problema deriva in particolare dall'incapacità della JVM di rilevare i limiti di memoria dei container utilizzando `cgroupv2`, l'impostazione predefinita in Amazon Linux 2023. Di conseguenza, basa l'allocazione della memoria heap sulla memoria totale del nodo anziché sul limite definito dal pod. La causa del problema è la modifica della posizione di archiviazione dei dati relativi al limite di memoria in `cgroupv2`, che causa la lettura errata della memoria disponibile e l'utilizzo di risorse a livello di nodo da parte delle versioni precedenti di Java. Ecco alcune possibili soluzioni:

- Aggiornare la versione del JDK: l'aggiornamento alla versione `jdk8u372` o successive o a una versione del JDK più recente con supporto completo di `cgroupv2` può risolvere questo problema. Per un elenco delle versioni Java compatibili con supporto completo di `cgroupv2`, consultare la pagina About cgroup v2.
- Crea un'AMI personalizzata: se devi continuare a utilizzare una AL2-based soluzione, puoi creare un' AL2-based AMI personalizzata (fino al 26 novembre 2025) utilizzando the section called "Crea un'AMI EKS-optimized Amazon Linux personalizzata" Amazon EKS AMI Build Specification. Ad esempio, puoi creare un' AL2-based AMI v1.33 (fino al 26 novembre 2025). Amazon EKS fornirà le

AL2-based AMI fino alla data EKS AL2 EOS (26 novembre 2025). Dopo la data di fine del supporto (26 novembre 2025), sarà necessario creare la propria AMI.

- Abilitare cgroupv1: se è necessario continuare a utilizzare `cgroupv1`, è possibile abilitar `cgroupv1` su un'AMI EKS AL2023. Per farlo, eseguire `sudo grubby --update-kernel=ALL --args="systemd.unified_cgroup_hierarchy=0"` e riavviare il sistema (ad esempio, istanza o nodo EC2 che esegue Amazon Linux 2023). Ciò modificherà i parametri di avvio del sistema (ad esempio, aggiungendo il parametro di kernel 'systemd.unified_cgroup_hierarchy=0', che indica a systemd di utilizzare la gerarchia legacy `cgroupv1`, alla configurazione GRUB) e abiliterà `cgroupv1`. Tenere presente che, con l'esecuzione di questo comando grubby, si procede alla configurazione del kernel in modo che venga avviato con `cgroupv1` abilitato e `cgroupv2` disabilitato. Per la gestione attiva delle risorse su un nodo viene utilizzata solo una di queste versioni di cgroup. Ciò non equivale all'esecuzione di `cgroupv2` con compatibilità retroattiva per l'API `cgroupv1`.

> **Warning**
>
> Sconsigliamo l'uso continuato di `cgroupv1`. Consigliamo invece di eseguire la migrazione a `cgroupv2`. La community di Kubernetes ha trasferito il supporto di `cgroupv1` (utilizzato da AL2) in modalità manutenzione, il che significa che non verranno aggiunti nuove funzionalità o aggiornamenti e verranno forniti solo gli aggiornamenti di sicurezza critici e le correzioni di bug più importanti. La rimozione completa del supporto di `cgroupv1` è prevista in una versione futura, anche se non è stata ancora annunciata una data specifica. Se riscontri problemi con`cgroupv1`, non AWS sarà in grado di fornire supporto e ti consigliamo di effettuare l'aggiornamento a. `cgroupv2`

## Compatibilità e versioni

### Versioni Kubernetes supportate per AMI AL2

La versione Kubernetes 1.32 è l'ultima per la quale Amazon EKS rilascerà AMI AL2 (Amazon Linux 2). Per le versioni di Kubernetes supportate fino alla 1.32, EKS continuerà a rilasciare AMI AL2 (AL2_ARM_64, AL2_x86_64) e AMI (AL2_x86_64_GPU) fino al 26 novembre 2025. AL2-accelerated Dopo questa data, AL2-optimized EKS AL2-accelerated smetterà di rilasciare AMI per tutte le versioni di Kubernetes. Tieni presente che la data EOS per EKS AL2-optimized e AL2-accelerated AMI è indipendente dalle tempistiche di supporto standard ed estese per le versioni Kubernetes di EKS.

Confronto delle versioni dei driver supportati e del kernel Linux per AMI AL2, AL2023 e Bottlerocket

| Componente | EX AL2 AMI | EX AL2023 AMI | EKS Bottlerocket AMI |
|---|---|---|---|
| Compatibilità con il sistema operativo di base | RHEL7/CentOS 7 | Fedora/CentOS 9 | N/A |
| Driver in modalità utente CUDA | 12.x | 12.x, 13.x | 12x, 13x |
| Driver GPU NVIDIA | R570 | R580 | R570, R580 |
| AWS Driver neuronale | 2,20 + | 2,20+ | 2,20+ |
| Kernel Linux | 5,10 | 6.1, 6.12 | 6.1, 6.12 |

Per ulteriori informazioni sui driver NVIDIA e sulla compatibilità con CUDA, consulta la documentazione NVIDIA.

AWS Compatibilità di Neuron con le AMI AL2

A partire dalla versione 2.20 di AWS Neuron, Neuron Runtime (`aws-neuronx-runtime-lib`) utilizzato dalle AL-based AMI EKS non supporta più Amazon Linux 2 (AL2). Il Neuron Driver (`aws-neuronx-dkms`) è ora l'unico pacchetto AWS Neuron che supporta Amazon Linux 2. Ciò significa che non è possibile eseguire le Neuron-powered applicazioni in modo nativo su un' AL2-based AMI. Per configurare Neuron sulle AMI AL2023, consultare la guida AWS Neuron Setup.

Compatibilità di Kubernetes con le AMI AL2

La community di Kubernetes ha trasferito il supporto di `cgroupv1` (utilizzato da AL2) in modalità manutenzione. Ciò significa che non verranno aggiunte nuove funzionalità e verranno forniti solo gli aggiornamenti di sicurezza critici e le correzioni di bug più importanti. Tutte le funzionalità di Kubernetes che si basano su cgroupv2, come MemoryQo S e l'isolamento avanzato delle risorse, non sono disponibili su AL2. Inoltre, la versione 1.32 di Amazon EKS Kubernetes è stata l'ultima versione a supportare le AMI AL2. Per mantenere la compatibilità con le versioni più recenti di Kubernetes, consigliamo di eseguire la migrazione ad AL2023 o Bottlerocket, che abilitano `cgroupv2` per impostazione predefinita.

### Compatibilità della versione Linux con le AMI AL2

Amazon Linux 2 (AL2) è supportato AWS fino alla data di fine del supporto (EOS), il 30 giugno 2026. Tuttavia, man mano che AL2 è diventato obsoleto, il supporto per nuove applicazioni e funzionalità da parte della community Linux più ampia è diventato più limitato. Le AMI AL2 sono basate sul kernel Linux 5.10, mentre AL2023 utilizza il kernel Linux 6.1. A differenza di AL2023, AL2 dispone di un supporto limitato da parte della community Linux più ampia. Ciò significa che molti pacchetti e strumenti Linux a monte devono essere sottoposti a backport per funzionare con la vecchia precedente del kernel di AL2, alcune funzionalità e miglioramenti della sicurezza moderni di Linux non sono disponibili a causa del kernel obsoleto, molti progetti open source hanno un supporto deprecato o limitato per le versioni precedenti del kernel come la 5.10.

### Pacchetti obsoleti non inclusi in AL2023

Di seguito sono elencati alcuni dei pacchetti più comuni che non sono inclusi o che sono stati modificati in AL2023:

- Alcuni pacchetti binari di origine in Amazon Linux 2 non sono più disponibili in Amazon Linux 2023
- Modifiche del supporto di Amazon Linux per versioni diverse dei pacchetti (ad esempio, il sistema amazon-linux-extras) in AL2023
- I pacchetti aggiuntivi per Enterprise Linux (EPEL) non sono supportati in AL2023
- Le applicazioni a 32 bit non sono supportate in AL2023

Per ulteriori informazioni, consultare la pagina Comparing AL2 and AL2023.

### Confronto della convalida FIPS tra AL2, AL2023 e Bottlerocket

Amazon Linux 2 (AL2), Amazon Linux 2023 (AL2023) e Bottlerocket offrono supporto per gli standard Federal Information Processing Standards (FIPS).

- AL2 è certificato secondo lo standard FIPS 140-2 e AL2023 è certificato secondo lo standard FIPS 140-3. Per abilitare la modalità FIPS su AL2023, installare i pacchetti necessari sull'istanza Amazon EC2 e seguire i passaggi di configurazione usando le istruzioni presenti alla pagina Enable FIPS Mode on AL2023. Per ulteriori informazioni, consultare la pagina di domande frequenti AL2023 FAQs.
- Bottlerocket offre varianti dedicate specificamente per FIPS che limitano i componenti del kernel e dello spazio utente all'uso di moduli crittografici che sono stati inviati al programma di convalida FIPS 140-3 Cryptographic Module Validation Program.

Changelog di driver e versioni delle AMI EKS

Per un elenco completo di tutti i componenti AMI EKS e delle relative versioni, consulta le note di rilascio dell'AMI di Amazon EKS su GitHub.

# Crea nodi con Amazon Linux ottimizzato AMIs

Amazon Elastic Kubernetes Service (Amazon EKS) fornisce Amazon Machine AMIs Images () specializzate ottimizzate per l'esecuzione di nodi di lavoro Kubernetes. Questi Amazon Linux (AL) ottimizzati per EKS AMIs sono preconfigurati con componenti essenziali, come AWS `kubelet` IAM Authenticator e, per garantire un'integrazione e `containerd` una sicurezza senza interruzioni all'interno dei cluster. Questa guida descrive in dettaglio le versioni AMI disponibili e delinea le opzioni specializzate per l'elaborazione accelerata e le architetture basate su ARM.

## Considerazioni

- Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux in Amazon Linux security center scegliendo la scheda per la versione desiderata. Puoi anche abbonarti al feed RSS applicabile. Gli eventi di sicurezza e privacy includono una panoramica del problema, quali sono i pacchetti interessati e come aggiornare le istanze per risolvere il problema.
- Prima di distribuire un'AMI accelerata o Arm, esamina le informazioni in Amazon Linux accelerato ottimizzato per Amazon EKS e. AMIs the section called “Arm ottimizzato per Amazon EKS Amazon Linux AMIs”
- EC2 `P2`Le istanze Amazon non sono supportate su Amazon EKS perché richiedono la versione 470 o precedente del `NVIDIA` driver.
- Tutti i gruppi di nodi gestiti appena creati nei cluster a partire dalla versione `1.30` più recente utilizzeranno automaticamente AL2 023 come sistema operativo del nodo.

## Amazon Linux accelerato ottimizzato per Amazon EKS AMIs

Gli Amazon Linux (AL AMIs ) accelerati ottimizzati per Amazon EKS si basano sullo standard Amazon Linux ottimizzato per EKS. AMIs Sono configurate per fungere da immagini facoltative per i nodi Amazon EKS per supportare i carichi di lavoro basati su GPU, Inferentia e Trainium.

Per ulteriori informazioni, consulta the section called “Usa le AMI GPU EKS Linux”.

## Arm ottimizzato per Amazon EKS Amazon Linux AMIs

Le istanze Arm offrono significativi risparmi sui costi per applicazioni aumentabili orizzontalmente e basate su Arm come server Web, microservizi containerizzati, flotte di memorizzazione nella cache e archivi di dati distribuiti. Quando si aggiungono nodi Arm al cluster, esaminare le considerazioni riportate di seguito.

- Se il cluster è stato implementato prima del 17 agosto 2020, è necessario eseguire un aggiornamento una tantum dei manifesti critici dei componenti aggiuntivi del cluster. In questo modo Kubernetes può estrarre l'immagine corretta per ogni architettura hardware utilizzata nel cluster. Per ulteriori informazioni sull'aggiornamento dei componenti aggiuntivi del cluster, consultare the section called "Passaggio 1: preparazione per l'aggiornamento". Se il cluster è stato implementato a partire dal 17 agosto 2020, il CoreDNS, `kube-proxy` e i plug-in della CNI di Amazon VPC per i componenti aggiuntivi di Kubernetes sono già predisposti alla multi-architettura.
- Le applicazioni distribuite nei nodi Arm devono essere compilate per Arm.
- Se li hai DaemonSets implementati in un cluster esistente o desideri distribuirli in un nuovo cluster in cui desideri distribuire anche nodi Arm, verifica che sia DaemonSet possibile eseguirli su tutte le architetture hardware del cluster.
- È possibile eseguire gruppi di nodi Arm e gruppi di nodi x86 nello stesso cluster. In tal caso, prendi in considerazione di implementare immagini del container multi-architettura in un repository del container come Amazon Elastic Container Registry, quindi aggiungi selettori di nodi ai manifesti in modo che Kubernetes sappia in quale architettura hardware può essere implementato un pod. Per ulteriori informazioni, consultare Inviare un'immagine multi-architettura nella Guida per l'utente di Amazon ECR ed il blog post Presentazione di immagini container multi-architettura per Amazon ECR.

## Ulteriori informazioni

Per ulteriori informazioni sull'uso di Amazon Linux ottimizzato per Amazon EKS AMIs, consulta le seguenti sezioni:

- Per utilizzare Amazon Linux con gruppi di nodi gestiti, consulta la sezione the section called "Gruppi di nodi gestiti".
- Per avviare nodi Amazon Linux autogestiti, consulta la sezione the section called "Ottieni le ultime novità IDs".
- Per informazioni sulla versione, consulta the section called "Ottenere informazioni sulla versione".

- Per recuperare la versione più recente IDs di Amazon AMIs Linux ottimizzata per Amazon EKS, consulta. the section called "Ottieni le ultime novità IDs"
- Per gli script open source utilizzati per creare applicazioni ottimizzate per Amazon EKS AMIs, consulta. the section called "Crea un'AMI EKS-optimized Amazon Linux personalizzata"

## Aggiornamento da Amazon Linux 2 ad Amazon Linux 2023

> **Warning**
>
> Amazon EKS ha interrotto la pubblicazione delle AMI EKS-optimized Amazon Linux 2 (AL2) il 26 novembre 2025. Le AMI basate su AL2023 e Bottlerocket per Amazon EKS sono disponibili per tutte le versioni di Kubernetes supportate, inclusa la 1.33 e successive.

AL2023 è un sistema Linux-based operativo progettato per fornire un ambiente sicuro, stabile e ad alte prestazioni per le tue applicazioni cloud. È la nuova generazione di Amazon Linux offerta da Amazon Web Services ed è disponibile in tutte le versioni di Amazon EKS supportate.

AL2023 offre diversi miglioramenti rispetto ad AL2. Per un confronto completo, consulta Comparing AL2 and Amazon Linux 2023 nella Guida per l'utente di Amazon Linux 2023. Diversi pacchetti sono stati aggiunti, aggiornati e rimossi da AL2. Si consiglia vivamente di testare le applicazioni con AL2023 prima dell'aggiornamento. Per un elenco di tutte le modifiche ai pacchetti in AL2023, consulta Package changes in Amazon Linux 2023 nelle Note di rilascio di Amazon Linux 2023.

Oltre a queste modifiche, devi tenere presente quanto riportato di seguito:

- AL2023 introduce un nuovo processo di inizializzazione dei nodi `nodeadm` che utilizza uno schema di configurazione YAML. Se utilizzi gruppi di nodi autogestiti o un modello di avvio che specifica un ID AMI personalizzato, ora devi fornire esplicitamente metadati aggiuntivi del cluster. È necessario fornire questi metadati quando si crea un nuovo gruppo di nodi. Di seguito è riportato un esempio dei parametri minimi richiesti da nodeadm `apiServerEndpointcertificateAuthority`, dove e il servizio `cidr` sono ora obbligatori:

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
```

```
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

In AL2, i metadati di questi parametri sono stati rilevati dalla chiamata `DescribeCluster` API di Amazon EKS. Con AL2023, questo comportamento è cambiato poiché la chiamata API aggiuntiva rischia una limitazione (della larghezza di banda della rete) durante gli aumenti verticali su larga scala dei nodi. Questa modifica non ha effetto su di te se utilizzi gruppi di nodi gestiti senza un modello di avvio o se utilizzi Karpenter. Per ulteriori informazioni su `certificateAuthority` e sul servizio `cidr`, consulta `DescribeCluster` nel Riferimento di API Amazon EKS.

- Per AL2023, `nodeadm` modifica anche il formato per applicare i parametri a `kubelet` per ogni nodo che utilizza `NodeConfigSpec`. In AL2, ciò è stato fatto con il parametro `--kubelet-extra-args`. Questo è usato comunemente per aggiungere etichette e taint ai nodi. L'esempio seguente mostra l'applicazione di `maxPods` e `--node-labels` al nodo.

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: test-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      maxPods: 110
    flags:
      - --node-labels=karpenter.sh/capacity-type=on-demand,karpenter.sh/nodepool=test
```

- Per AL2023 è richiesta la versione del CNI di Amazon VPC `1.16.2` o superiore.
- AL2023 richiede `IMDSv2` per impostazione predefinita. `IMDSv2` presenta diversi vantaggi che aiutano a migliorare il livello di sicurezza. Usa un metodo di autenticazione orientato alla sessione che prevede la creazione di un token segreto in una semplice richiesta HTTP PUT per avviare la sessione. Il token di una sessione può avere una validità compresa tra 1 secondo e 6 ore. Per ulteriori informazioni su come passare da a`IMDSv2`, consulta Transition `IMDSv1` to using Instance Metadata Service Version 2 e Ottieni tutti i vantaggi di IMDSv2 e disabilita IMDSv1 nella tua infrastruttura. AWS Se desideri usare `IMDSv1`, puoi farlo comunque sovrascrivendo manualmente le impostazioni usando le proprietà di avvio dell'opzione dei metadati di istanza.

> **Note**
>
> Per `IMDSv2` con AL2023, il numero di hop predefinito per i gruppi di nodi gestiti può variare:
>
> - Quando non si utilizza un modello di avvio, il valore predefinito è impostato su `1`. Ciò significa che i container non avranno accesso alle credenziali del nodo che usano IMDS. Se necessiti dell'accesso del container alle credenziali del nodo, puoi comunque farlo utilizzando un modello di lancio Amazon EC2 personalizzato.
> - Quando si utilizza un'AMI personalizzata in un modello di avvio, il valore predefinito `HttpPutResponseHopLimit` è impostato su `2`. Puoi sostituire manualmente `HttpPutResponseHopLimit` nel modello di avvio.
>
> In alternativa, puoi utilizzare Amazon EKS Pod Identity per fornire credenziali anziché `IMDSv2`.

- AL2023 presenta la nuova generazione di gerarchia unificata dei gruppi di controllo (`cgroupv2`). `cgroupv2` viene utilizzato per implementare un runtime di container e da parte di `systemd`. Sebbene AL2023 contenga ancora codice che consentirebbe il funzionamento del sistema con `cgroupv1`, si tratta di una configurazione sconsigliata o non supportata. Questa configurazione verrà completamente rimossa in una delle future release principali di Amazon Linux.
- `eksctl` versione `0.176.0` o successiva è richiesta affinché `eksctl` supporti AL2023.

Per i gruppi di nodi gestiti precedentemente esistenti, puoi eseguire un aggiornamento sul posto o un aggiornamento a seconda di come stai utilizzando un modello di blue/green avvio:

- Se utilizzi un'AMI personalizzata con un gruppo di nodi gestito, puoi eseguire un aggiornamento sul posto scambiando l'ID AMI nel modello di avvio. Devi assicurarti che le applicazioni e tutti i dati utente vengano trasferiti ad AL2023 prima di eseguire questa strategia di aggiornamento.
- Se utilizzi gruppi di nodi gestiti con il modello di avvio standard o con un modello di avvio personalizzato che non specifica l'ID AMI, devi eseguire l'aggiornamento utilizzando una blue/green strategia. Un blue/green aggiornamento è in genere più complesso e comporta la creazione di un gruppo di nodi completamente nuovo in cui è necessario specificare AL2023 come tipo di AMI. Il nuovo gruppo di nodi dovrà quindi essere configurato con cura per garantire che tutti i dati personalizzati del gruppo di nodi AL2 siano compatibili con il nuovo sistema operativo. Una volta che il nuovo gruppo di nodi è stato testato e convalidato con le applicazioni, i Pod possono essere

migrati dal vecchio gruppo di nodi al nuovo gruppo di nodi. Una volta completata la migrazione, puoi eliminare il vecchio gruppo di nodi.

Se utilizzi Karpenter e desideri utilizzare AL2023, dovrai modificare il campo `EC2NodeClass` `amiFamily` con AL2023. Per impostazione predefinita, Drift è abilitato in Karpenter. Ciò significa che una volta modificato il campo `amiFamily`, Karpenter aggiornerà automaticamente i nodi worker all'AMI più recente, quando disponibile.

Informazioni aggiuntive su nodeadm

Quando utilizzi un'AMI EKS-optimized Amazon Linux 2023 o crei un'AMI Amazon Linux 2023 personalizzata con gli script Packer forniti nel GitHub repository ufficiale amazon-eks-ami, dovresti evitare di eseguire esplicitamente nodeadm init all'interno dei dati utente di EC2 o come parte della tua AMI personalizzata.

Se desideri generare una dinamica NodeConfig nei tuoi dati utente, puoi scrivere tale configurazione in un file yaml o json drop-in. `/etc/eks/nodeadm.d` Questi file di configurazione verranno uniti e applicati al nodo quando nodeadm init viene avviato automaticamente più avanti nel processo di avvio. Ad esempio:

```
cat > /etc/eks/nodeadm.d/additional-node-labels.yaml << EOF
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  kubelet:
    flags:
      - --node-labels=foo=bar
EOF
```

Le AMI di EKS-optimized Amazon Linux 2023 eseguono automaticamente nodeadm init in due fasi tramite servizi systemd separati: nodeadm-config viene eseguito prima dell'esecuzione dei dati utente, mentre nodeadm-run viene eseguito successivamente. Il servizio nodeadm-config stabilisce le configurazioni di base per containerd e kubelet prima dell'esecuzione dei dati utente. Il servizio nodeadm-run esegue alcuni demoni di sistema e completa tutte le configurazioni finali dopo l'esecuzione dei dati utente. Se il comando nodeadm init viene eseguito un'altra volta, tramite dati utente o un'AMI personalizzata, può infrangere le ipotesi sull'ordine di esecuzione, portando a risultati imprevisti, tra cui ENI non configurati correttamente.

## Recupero delle informazioni sulla versione dell'AMI Amazon Linux

Le versioni delle AMI Amazon Linux ottimizzate per Amazon EKS si basano sulla versione di Kubernetes e la data di rilascio dell'AMI è nel formato seguente:

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

Ogni rilascio dell'AMI include varie versioni di kubelet, del kernel Linux e di containerd. L'AMI accelerata include anche varie versioni del driver NVIDIA. Puoi trovare queste informazioni sulla versione nella sezione Releases on GitHub.

## Recupera le AMI Amazon Linux consigliate IDs

Durante la distribuzione dei nodi, è possibile specificare un ID per un'Amazon Machine Image (AMI) preconfigurata e ottimizzata per Amazon EKS. Per recuperare un ID AMI adatto alla configurazione desiderata, interroga l'API AWS Systems Manager Parameter Store. L'utilizzo di questa API elimina la necessità di cercare manualmente le AMI ottimizzate per Amazon EKS IDs. Per ulteriori informazioni, consulta GetParameter. Il principale IAM che usi deve disporre dell'autorizzazione IAM `ssm:GetParameter` per recuperare i metadati dell'AMI ottimizzata per Amazon EKS.

È possibile recuperare l'ID immagine dell'ultima versione consigliata dell'AMI Amazon Linux ottimizzata per Amazon EKS con il comando seguente, che utilizza il parametro secondario `image_id`. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

- Sostituisci <kubernetes-version> con una versione supportata da Amazon EKS.
- Sostituisci *ami-type* con una delle seguenti opzioni. Per informazioni sui tipi di EC2 istanze Amazon, consulta Tipi di EC2 istanze Amazon.
  - Utilizzalo *amazon-linux-2023/x86_64/standard* per istanze x86 basate su Amazon Linux 2023 (AL2023).
  - Utilizzalo *amazon-linux-2023/arm64/standard* per AL2 023 istanze ARM, come AWS le istanze basate su Graviton.
  - Da utilizzare *amazon-linux-2023/x86_64/nvidia* per le ultime AL2 023 istanze basate su NVIDIA approvate. x86
  - Da utilizzare *amazon-linux-2023/arm64/nvidia* per le ultime istanze AL2 023 basate su NVIDIA approvate. arm64

  - Da utilizzare *amazon-linux-2023/x86_64/neuron* per le ultime istanze AL2 AWS 023 Neuron.
- Sostituisci `<region-code>` con una AWS regione supportata da Amazon EKS per la quale desideri l'ID AMI.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/<kubernetes-version>/<ami-type>/recommended/image_id \
    --region <region-code> --query "Parameter.Value" --output text
```

Ecco un comando esemplificativo dopo la sostituzione dei segnaposto.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.31/amazon-linux-2023/x86_64/standard/recommended/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Di seguito viene riportato un output di esempio.

```
ami-1234567890abcdef0
```

## Crea un'AMI EKS-optimized Amazon Linux personalizzata

> **Warning**
>
> Amazon EKS ha smesso di pubblicare AMI EKS-optimized Amazon Linux 2 (AL2) il 26 novembre 2025. Le AMI basate su AL2023 e Bottlerocket per Amazon EKS sono disponibili per tutte le versioni di Kubernetes supportate, inclusa la 1.33 e successive.

Amazon EKS fornisce script di build open source nel repository Amazon EKS AMI Build Specification che puoi usare per visualizzare le configurazioni, il runtime`kubelet`, l' AWS IAM Authenticator per Kubernetes e creare la tua AMI da zero. AL-based

Questo repository contiene lo script bootstrap specializzato per AL2 e lo strumento nodeadm per AL2023 che viene eseguito all'avvio. Questi script configurano i dati del certificato dell'istanza, l'endpoint del piano di controllo (control-plane), il nome del cluster e altro ancora. Gli script sono considerati la fonte di verità per le build di Amazon EKS-optimized AMI, quindi puoi seguire il GitHub repository per monitorare le modifiche alle nostre AMI.

Quando si creano AMI personalizzate con le EKS-optimized AMI come base, non è consigliato né supportato l'esecuzione di un aggiornamento del sistema operativo (ad es. `dnf upgrade`) o aggiorna uno qualsiasi dei pacchetti Kubernetes o GPU inclusi nelle AMI, poiché ciò rischia di compromettere la EKS-optimized compatibilità dei componenti. Se si esegue l'aggiornamento del sistema operativo o dei pacchetti inclusi nelle EKS-optimized AMI, si consiglia di eseguire test approfonditi in un ambiente di sviluppo o di gestione temporanea prima di passare alla produzione.

Quando si creano AMI personalizzate per istanze GPU, si consiglia di creare AMI personalizzate separate per ogni tipo di generazione e famiglia di istanze che verrà eseguita. Le AMI EKS-optimized accelerate installano selettivamente driver e pacchetti in fase di esecuzione in base alla generazione e alla famiglia di istanze sottostanti. Per ulteriori informazioni, consulta gli script EKS AMI per l'installazione e il runtime.

Prerequisiti

- Installa la AWS CLI
- Installa HashiCorp Packer v1.9.4+
- Installare GNU Make

Installare il plugin Amazon richiesto.

```
packer plugins install github.com/hashicorp/amazon
```

Verifica che Packer sia installato.

```
packer --version
```

Clonare o creare un fork del repository Amazon EKS AMI ufficiale.

```
git clone https://github.com/awslabs/amazon-eks-ami.git
cd amazon-eks-ami
```

Per visualizzare i valori predefiniti e le opzioni aggiuntive, esegui il comando seguente. Per ulteriori informazioni sulle configurazioni disponibili per personalizzare l'AMI, consultare le variabili del modello alla pagina Amazon Linux 2023.

```
make help
```

Creazione di un'AMI personalizzata

Di seguito sono riportati esempi di comandi per diverse AMI personalizzate.

AMI EKS NVIDIA AL2

```
make k8s=1.32 os_distro=al2 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=580 \
  enable_efa=true
```

AMI EKS NVIDIA AL2023

```
make k8s=1.36 os_distro=al2023 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=580 \
  enable_efa=true
```

AMI EKS NVIDIA AL2023 con driver NVIDIA versione 595

> **Important**
>
> Il tipo di istanza G7 EC2 richiede il driver NVIDIA versione 595 o successiva. Le AMI EKS-optimized accelerate attualmente includono la versione 580 del driver NVIDIA, che non supporta le istanze G7. È necessario creare un'AMI personalizzata con il driver NVIDIA versione 595 per utilizzare le istanze G7 con Amazon EKS.
> Se utilizzi Karpenter per il provisioning e l'auto-scaling dei nodi, ti consigliamo di escludere la famiglia di g7 istanze che utilizza la selezione automatica delle AMI. NodePools `amiSelectorTerms`Consulta la documentazione di Karpenter per scoprire come configurare Karpenter per utilizzare l'AMI personalizzata NodeClass per le istanze. g7

```
make k8s=1.36 os_distro=al2023 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=595 \
  enable_efa=true
```

STIG-Compliant AMI neuronale AL2023:

```
make k8s=1.31 os_distro=al2023 \
```

```
  enable_accelerator=neuron \
  enable_fips=true \
  source_ami_id=ami-0abcd1234efgh5678 \
  kms_key_id=alias/aws-stig
```

Dopo aver eseguito questi comandi, Packer eseguirà le seguenti operazioni:

- Avvia un'istanza Amazon EC2 temporanea.
- Installa componenti, driver e configurazioni Kubernetes.
- Crea l'AMI nel tuo AWS account.

L'output previsto dovrebbe essere simile al seguente:

```
==> Wait completed after 8 minutes 42 seconds

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9

--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9

--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9
```

## Creazione di nodi con AMI Bottlerocket ottimizzate

Bottlerocket è una distribuzione Linux open source sponsorizzata e supportata da. AWS Bottlerocket è progettato appositamente per ospitare carichi di lavoro in container. Con Bottlerocket, puoi migliorare la disponibilità delle implementazioni containerizzate e ridurre i costi operativi automatizzando gli aggiornamenti dell'infrastruttura dei container. Bottlerocket include solo il software essenziale per eseguire i container, che migliora l'utilizzo delle risorse, riduce le minacce alla sicurezza e riduce i costi di gestione. L'AMI Bottlerocket include e AWS IAM `containerd` `kubelet` Authenticator. Oltre che dai gruppi di nodi gestiti e di nodi autogestiti, Bottlerocket è supportato anche da Karpenter.

### Vantaggi

L'utilizzo di Bottlerocket con il cluster Amazon EKS presenta i seguenti vantaggi:

- Tempi di attività maggiori con costi operativi inferiori e minore complessità di gestione: Bottlerocket necessita di un minore utilizzo di risorse, tempi di avvio più brevi ed è meno vulnerabile alle minacce alla sicurezza rispetto ad altre distribuzioni Linux. L'ingombro ridotto di Bottlerocket aiuta a ridurre i costi utilizzando meno risorse di archiviazione, elaborazione e rete.
- Maggiore sicurezza grazie agli aggiornamenti automatici del sistema operativo – gli aggiornamenti a Bottlerocket vengono applicati come una singola unità che può essere ripristinata, se necessario. Ciò elimina il rischio di aggiornamenti danneggiati o non riusciti che possono lasciare il sistema in uno stato inutilizzabile. Con Bottlerocket, gli aggiornamenti di sicurezza possono essere applicati automaticamente non appena sono disponibili con un impatto minimo e possono essere ripristinati in caso di guasti.
- Supporto premium: le build di Bottlerocket AWS fornite su Amazon EC2 sono coperte dagli stessi piani di AWS supporto che coprono anche servizi AWS come Amazon EC2, Amazon EKS e Amazon ECR.

## Considerazioni

Quando utilizzi Bottlerocket per il tuo tipo di AMI, considera quanto segue:

- Bottlerocket supporta le istanze Amazon EC2 con processori `x86_64` e `arm64`.
- Bottlerocket supporta istanze Amazon EC2 con GPU. Per ulteriori informazioni, consulta the section called "Usa le AMI GPU EKS Linux".
- Le immagini Bottlerocket non includono un server SSH o una shell. Puoi utilizzare metodi di accesso fuori banda per consentire SSH. Questi approcci consentono di utilizzare il container dell'amministratore e saltare alcune fasi di configurazione di bootstrap con i dati utente. Per ulteriori informazioni, consulta le seguenti sezioni di Bottlerocket OS su: GitHub
  - Esplorazione
  - Container amministratore
  - Impostazioni Kubernetes
- Bottlerocket utilizza diversi tipi di container:
  - Per impostazione predefinita, è abilitato un container di controllo. Questo container gestisce l'agente AWS Systems Manager che è possibile utilizzare per eseguire comandi o avviare sessioni di shell sulle istanze Bottlerocket di Amazon EC2. Per ulteriori informazioni, vedere Configurazione di Session Manager nella Guida per l'utente di AWS Systems Manager.
  - Se viene fornita una chiave SSH durante la creazione del gruppo di nodi, viene abilitato un container dell'amministratore. Consigliamo di utilizzare il container amministratore solo per

scenari di sviluppo e test. Non è consigliabile utilizzarlo in ambienti di produzione. Per ulteriori informazioni, consulta Container amministratore su GitHub.

## Ulteriori informazioni

Per ulteriori informazioni sull'utilizzo delle AMI Bottlerocket ottimizzate per Amazon EKS, consulta le sezioni seguenti:

- Per i dettagli su Bottlerocket, consulta la documentazione di Bottlerocket.
- Per le risorse informative sulla versione, consulta the section called "Ottenere informazioni sulla versione".
- Per utilizzare Bottlerocket con gruppi di nodi gestiti, consulta the section called "Gruppi di nodi gestiti".
- Per avviare i nodi autogestiti di Bottlerocket, consulta the section called "Razzo per bottiglie".
- Per recuperare gli ID più recenti delle AMI Bottlerocket ottimizzate per Amazon EKS, consulta the section called "Ottenimento degli ID della versione più recente".
- Per informazioni dettagliate sul supporto per la conformità, consulta la sezione the section called "Supporto alla conformità".
- Per creare una variante Bottlerocket personalizzata con un driver NVIDIA specifico, consulta. the section called "Crea un'AMI Bottlerocket personalizzata"

## Recupero delle informazioni sulla versione delle AMI Bottlerocket

Ogni versione delle AMI Bottlerocket include varie versioni di kubelet, del kernel Bottlerocket e di containerd. Le versioni dell'AMI accelerata includono anche varie versioni del driver NVIDIA. È possibile trovare queste informazioni sulla versione nell'argomento relativo al sistema operativo OS della documentazione di Bottlerocket. Da questa pagina, andare all'argomento secondario Version Information applicabile.

A volte la documentazione di Bottlerocket può essere in ritardo rispetto alle versioni disponibili su GitHub. È possibile trovare un elenco di modifiche relative alle versioni più recenti alla pagina releases su GitHub.

## Recupero degli ID AMI Bottlerocket consigliati

Durante la distribuzione dei nodi, è possibile specificare un ID per un'Amazon Machine Image (AMI) preconfigurata e ottimizzata per Amazon EKS. Per recuperare un ID AMI adatto alla configurazione

desiderata, interroga l'API AWS Systems Manager Parameter Store. L'utilizzo di tale parametro consente di evitare la ricerca manuale degli ID AMI ottimizzata per Amazon EKS. Per ulteriori informazioni, consulta Ottenere parametro. Il principale IAM che usi deve disporre dell'autorizzazione IAM `ssm:GetParameter` per recuperare i metadati dell'AMI ottimizzata per Amazon EKS.

È possibile recuperare l'ID immagine della versione più recente dell'AMI Bottlerocket ottimizzata per Amazon EKS consigliata con il seguente comando AWS CLI, che utilizza il parametro secondario `image_id`. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

- Sostituisci la *`versione Kubernetes`* con una versione della piattaforma supportata.
- Sostituisci *`-flavor`* con una delle seguenti opzioni.
  - Rimuovi *`-flavor`* per le varianti senza GPU.
  - Utilizza *`-nvidia`* per le varianti abilitate per la GPU.
  - Utilizza *`-fips`* per le varianti compatibili con FIPS.
- Sostituisci *`architettura`* con una delle seguenti opzioni.
  - Utilizza *`x86_64`* per le istanze basate su x86.
  - Utilizza *`arm64`* per le istanze ARM.
- Sostituisci *`codice regione`* con una regione AWS supportata da Amazon EKS per la quale si desidera l'ID AMI.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-kubernetes-version-
flavor/architecture/latest/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Ecco un comando esemplificativo dopo la sostituzione dei segnaposto.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.31/x86_64/latest/
image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Di seguito viene riportato un output di esempio:

```
ami-1234567890abcdef0
```

## Soddisfare i requisiti di conformità con Bottlerocket

Bottlerocket è conforme alle raccomandazioni definite da varie organizzazioni:

- È stato definito un Benchmark CIS per Bottlerocket. In una configurazione predefinita, l'immagine Bottlerocket presenta la maggior parte dei controlli richiesti dal profilo di configurazione CIS Level 1. Puoi implementare i controlli necessari per un profilo di configurazione CIS di livello 2. Per ulteriori informazioni, consulta Convalida dell'AMI Bottlerocket ottimizzata per Amazon EKS rispetto al benchmark CIS sul blog. AWS
- Il set di funzionalità ottimizzato e la superficie di attacco ridotta fanno sì che le istanze Bottlerocket richiedano meno configurazioni per soddisfare i requisiti PCI DSS. Il Benchmark CIS per Bottlerocket è una risorsa eccellente per le linee guida in materia di hardening e supporta i requisiti per gli standard di configurazione sicuri ai sensi del requisito PCI DSS 2.2. Puoi anche sfruttare Fluent Bit per soddisfare i tuoi requisiti di registrazione degli audit a livello di sistema operativo secondo il requisito PCI DSS 10.2. AWS pubblica periodicamente nuove istanze Bottlerocket (con patch) per aiutarvi a soddisfare il requisito PCI DSS 6.2 (per la v3.2.1) e il requisito 6.3.3 (per la v4.0).
- Bottlerocket è una HIPAA-eligible funzionalità autorizzata per l'uso con carichi di lavoro regolamentati sia per Amazon EC2 che per Amazon EKS. Per ulteriori informazioni, consulta HIPAA Eligible Services Reference.
- Sono disponibili AMI Bottlerocket preconfigurate per utilizzare moduli crittografici convalidati FIPS 140-3. Ciò include il modulo crittografico Amazon Linux 2023 Kernel Crypto API e il modulo crittografico AWS-LC. Per ulteriori informazioni, consulta the section called “AMI Bottlerocket FIPS”.

## Rendi i tuoi nodi di lavoro compatibili con il FIPS grazie alle AMI FIPS Bottlerocket

La pubblicazione 140-3 del Federal Information Processing Standard (FIPS) è uno standard di sicurezza dei governi degli Stati Uniti e del Canada che specifica i requisiti di sicurezza previsti per i moduli crittografici che proteggono informazioni sensibili. Bottlerocket semplifica la conformità allo standard FIPS offrendo AMI con kernel FIPS.

Queste AMI sono preconfigurate per usare moduli crittografici convalidati FIPS 140-3. Ciò include il modulo crittografico Amazon Linux 2023 Kernel Crypto API e il modulo crittografico Go.

L'uso delle AMI FIPS Bottlerocket rende i nodi di lavoro «pronti per FIPS» ma non automaticamente «». FIPS-compliant Per ulteriori informazioni, consulta Federal Information Processing Standard (FIPS) 140-3.

Considerazioni

- Se il cluster usa sottoreti isolate, l'endpoint FIPS di Amazon ECR potrebbe non essere accessibile. Questo può causare il fallimento del bootstrap del nodo. Assicurati che la configurazione di rete consenta l'accesso agli endpoint FIPS necessari. Per ulteriori informazioni, consulta Accedere a una risorsa tramite un endpoint VPC di risorse nella Guida. AWS PrivateLink
- Se il tuo cluster utilizza una sottorete con PrivateLink, assicurati di aver configurato gli endpoint PrivateLink FIPS di Amazon ECR. Amazon ECR supporta gli endpoint FIPS tramite. PrivateLink Per ulteriori informazioni, consulta Endpoint VPC con interfaccia Amazon ECR.

Creare un gruppo di nodi gestiti con un'AMI Bottlerocket FIPS

L'AMI FIPS Bottlerocket è disponibile in quattro varianti per supportare i tuoi carichi di lavoro:

- `BOTTLEROCKET_x86_64_FIPS`
- `BOTTLEROCKET_ARM_64_FIPS`
- `BOTTLEROCKET_x86_64_NVIDIA_FIPS`
- `BOTTLEROCKET_ARM_64_NVIDIA_FIPS`

Per creare un gruppo di nodi gestiti con un'AMI FIPS Bottlerocket, seleziona il tipo di AMI applicabile durante il processo di creazione. Per ulteriori informazioni, consulta the section called "Crea".

Per ulteriori informazioni sulla selezione FIPS-enabled delle varianti, consulta. the section called "Ottenimento degli ID della versione più recente"

Disabilita l'endpoint FIPS per i dispositivi non supportati AWS Regioni

Le AMI FIPS Bottlerocket sono supportate direttamente negli Stati Uniti d'America, incluse AWS GovCloud le regioni (Stati Uniti). Per AWS le regioni in cui le AMI sono disponibili ma non sono supportate direttamente, puoi comunque utilizzare le AMI creando un gruppo di nodi gestito con un modello di avvio.

L'AMI FIPS Bottlerocket si basa sull'endpoint FIPS Amazon ECR durante il bootstrap, che generalmente non è disponibile al di fuori degli Stati Uniti. Per utilizzare l'AMI per il suo kernel FIPS in AWS regioni in cui non è disponibile l'endpoint FIPS Amazon ECR, procedi nel seguente modo per disabilitare l'endpoint FIPS:

1. Crea un nuovo file di configurazione con il contenuto seguente o incorpora il contenuto nel file di configurazione esistente.

```
[default]
use_fips_endpoint=false
```

1. Codifica il contenuto del file nel formato Base64.
2. Nel campo `UserData` del modello di avvio, aggiungi la seguente stringa codificata usando il formato TOML:

```
[settings.aws]
config = "<your-base64-encoded-string>"
```

Per altre impostazioni, consulta la descrizione delle impostazioni su di Bottlerocket. GitHub

Ecco un esempio di `UserData` in un modello di avvio:

```
[settings]
motd = "Hello from eksctl!"
[settings.aws]
config = "W2RlZmF1bHRdCnVzZV9maXBzX2VuZHBvaW50PWZhbHNlCg==" # Base64-encoded string.
[settings.kubernetes]
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
cluster-name = "<cluster-name>"
...<other-settings>
```

Per ulteriori informazioni sulla creazione di un modello di avvio con i dati utente, consulta the section called "Modelli di avvio".

## Crea una variante AMI Bottlerocket personalizzata per Amazon EKS

Quando esegui carichi di lavoro GPU su Amazon Elastic Kubernetes Service (Amazon EKS), scegli una variante Bottlerocket. La variante deve corrispondere alla versione di Kubernetes e al tipo di acceleratore in uso. Bottlerocket fornisce varianti convalidate per configurazioni comuni. Tuttavia, la tua organizzazione potrebbe aver bisogno di una variante diversa per uno dei seguenti motivi:

- Un ramo di driver NVIDIA più recente

- Una versione del driver bloccata per la conformità alle normative
- Pacchetti aggiuntivi per il monitoraggio
- Una linea di base consolidata richiesta dal team di sicurezza

Poiché Bottlerocket sul GitHub sito Web è completamente open source, puoi creare una variante personalizzata che soddisfi queste esigenze. Questo argomento mostra come clonare una variante esistente e sostituire il driver NVIDIA dal ramo R580 a R595 (versione 595.71.05). Quindi crei l'immagine e la registri come AMI privata.

> **Important**
>
> Il tipo di istanza G7 EC2 richiede il driver NVIDIA versione 595 o successiva. Le AMI NVIDIA EKS Bottlerocket attualmente includono il driver NVIDIA versione 580, che non supporta le istanze G7.
> Consulta il repository Bottlerocket sul GitHub sito Web per istruzioni su come creare una variante con il driver NVIDIA versione 595. Questo argomento illustra l'intero processo a partire dalla Fase 1.

Come funziona il sistema di compilazione

Quando corri`cargo make -e BUILDSYS_VARIANT=aws-k8s-1.36-nvidia`, succedono tre cose:

- Recupera le dipendenze. Twoliter (l'orchestrator di build di Bottlerocket) legge `Twoliter.toml` ed estrae tre artefatti della Open Container Initiative (OCI) da: `public.ecr.aws/bottlerocket`
  - bottlerocket-sdk: un'immagine del contenitore con la toolchain completa di compilazione incrociata (macro GCC, Rust, Go, RPM).
  - bottlerocket-kernel-kit: RPM predefiniti per kernel, moduli kernel (inclusi i pacchetti NVIDIA kmod) e firmware.
  - bottlerocket-core-kit — RPM predefiniti per userspace: kubelet, containerd, il plug-in per dispositivi NVIDIA e il toolkit contenitore, plug-in di impostazioni e servizi di sistema.

    `Twoliter.toml`blocca `Twoliter.lock` le versioni e blocca i relativi digest. Per cambiarle, `./tools/twoliter/twoliter update` esegui per ri-risolvere.

- Costruisci la variante. Twoliter lancia una build Docker all'interno del contenitore SDK. Compila la cartella settings-defaults della variante, risolve l'albero delle dipendenze RPM dai kit e assembla tutto in un'immagine del disco.
- Uscita. Twoliter scrive il `.img.lz4` file finale su. `build/images/` La build produce un output deterministico: gli stessi `Twoliter.toml` pin e la stessa variante generano `Cargo.toml` sempre la stessa immagine, indipendentemente dall'host.

Layout del repository

Le seguenti directory sono rilevanti per il lavoro sulle varianti:

```
bottlerocket/
### Cargo.toml                      # workspace: lists every variant
### Twoliter.toml                   # pins SDK + kit versions
### Twoliter.lock                   # locked digests for the above
### Licenses.toml                   # you create this (NVIDIA license acknowledgement)
### Infra.toml                      # you create this (AMI publish regions)
#
### variants/
#   ### aws-k8s-1.36-nvidia/       # example variant you'll copy
#   #   ### Cargo.toml             #   package list + kernel params
#   #   ### amispec.toml           #   symlink # ../shared/amispec-split.toml
#   ### shared/                    # shared AMI spec templates
#
### sources/
#   ### Cargo.toml                 # workspace: lists every settings-defaults crate
#   ### shared-defaults/           # the actual defaults (symlink targets)
#   ### settings-defaults/
#       ### aws-k8s-1.36-nvidia/
#           ### Cargo.toml
#           ### defaults.d/        # 30+ symlinks into shared-defaults/
#
### packages/
    ### settings-defaults/
    #   ### settings-defaults.spec   # RPM: declares which variants exist
    ### settings-plugins/
        ### settings-plugins.spec    # RPM: maps variants to settings plugins
```

Leggi le seguenti note sulla struttura del repository:

- Una variante è costituita principalmente da metadati. I kit esterni forniscono il kernel, i driver e lo spazio utente.
- Settings-defaults i file sono collegamenti simbolici, non copie. Usa `cp -R` (non `cp -r` su macOS) per conservarli.
- L'aggiunta di una variante richiede modifiche in cinque punti: due file dell'area di lavoro, due `Cargo.toml` `.spec` file e. `README.md`

Prerequisiti

Per completare la procedura dettagliata, è necessario:

- Un AWS account con le autorizzazioni per avviare istanze EC2 e registrare AMI
- Un'istanza EC2 (o un host Linux x86_64 equivalente) con almeno 8 core, 16 GiB di memoria e 150 GB di disco
- Ubuntu 24.04 LTS (o Fedora; macOS non è supportato come host di build)
- Docker 20.10 o successivo
- Rust (toolchain stabile, installata tramite rustup)
- cargo-make (ultima versione)
- Familiarità con i concetti di packaging Git, Rust's Cargo e RPM

> **Note**
>
> Una volta completata questa procedura dettagliata, interrompi l'istanza EC2 e annulla la registrazione di tutte le AMI che non ti servono più per evitare addebiti continui. Per istruzioni sulla pulizia, consulta. the section called “Pulizia”

Fase 1: Preparare l'host di compilazione

Avvia un'istanza EC2, ad esempio una `c7i.8xlarge` (32 vCPU, 64 GiB di memoria). Utilizza un volume root gp3 da 150 GB e allega la policy `AmazonSSMManagedInstanceCore` gestita per l'accesso SSM.

Connect all'istanza utilizzando AWS Systems Manager (SSM) Session Manager (SSM):

```
aws ssm start-session --target <instance-id>
```

```
cd ~
```

Installa i pacchetti del sistema operativo richiesti:

```
apt-get update
apt-get install -y build-essential openssl libssl-dev pkg-config lz4 \
                   git ca-certificates curl gnupg
```

> **Note**
>
> I `BUILDING.md` riferimenti ufficiali`liblz4-tool`. Nelle versioni recenti di Ubuntu, il pacchetto è denominato`lz4`.

Installa Docker usando i seguenti comandi:

```
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
  https://download.docker.com/linux/ubuntu noble stable" \
  > /etc/apt/sources.list.d/docker.list
apt-get update
apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
systemctl enable --now docker
```

Installa Rust e cargo-make usando i seguenti comandi:

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh -s -- -y
. "$HOME/.cargo/env"
cargo install cargo-make
```

Passaggio 2: clonare il repository

Clona il repository Bottlerocket sul GitHub sito Web e accedi alla directory:

```
cd ~/bottlerocket
```

Per creare una build riproducibile, dai un'occhiata a una versione con tag (ad esempio,). `git checkout v1.62.1` Per utilizzare i pacchetti più recenti, rimanete nella filiale. `develop`

Fase 3: Verifica le versioni del kit

Apri `Twoliter.toml` e conferma le versioni del kit. Per il supporto di R595, è necessaria la `bottlerocket-kernel-kit` versione 6.2.2 o successiva:

```
[[kit]]
name = "bottlerocket-kernel-kit"
version = "6.2.2"
vendor = "bottlerocket"
```

Se la versione è precedente, aggiornala e rigenera il lucchetto:

```
./tools/twoliter/twoliter update
```

Fase 4: Risolvi il problema del percorso di fabbricazione del carico

Twoliter è impostato `CARGO_HOME` su`~/bottlerocket/.cargo`, il che impedisce al processo interno di Cargo di trovare la marca di carico installata a livello globale. Crea un symlink:

```
mkdir -p ~/bottlerocket/.cargo/bin
ln -sf /root/.cargo/bin/cargo-make ~/bottlerocket/.cargo/bin/cargo-make
```

Senza questo passaggio, la build fallisce con. `error: no such command: make`

Passaggio 5: Trova le filiali di driver disponibili

I pacchetti mod NVIDIA vengono spediti all'interno. `bottlerocket-kernel-kit` Per informazioni sui pacchetti driver disponibili, consulta la directory dei pacchetti kernel-kit sul sito Web o le note di rilascio sul GitHub sito Web. GitHub

Per il kernel 6.18 (usato dalle varianti): `aws-k8s-1.36`

```
kmod-6.18-nvidia-r580   # driver 580.159.03 (current default)
kmod-6.18-nvidia-r595   # driver 595.71.05
```

Per il kernel 6.12 (usato da`aws-k8s-1.33`,,): `1.34` `1.35`

```
kmod-6.12-nvidia-r580
kmod-6.12-nvidia-r595
```

> **Note**
>
> Ogni pacchetto kmod spedisce pacchetti secondari (`-tesla`,,,`-open-gpu`,`-grid`). `-fabricmanager` `-imex` Il riferimento ufficiale alle varianti NVIDIA di Bottlerocket`-tesla`, che richiama tutti i sottopacchetti richiesti tramite dipendenze RPM. All'avvio, Bottlerocket seleziona automaticamente il tipo di driver appropriato in base al tipo di istanza.

Fase 6: Creare la nuova variante

Copia la variante esistente. Usa `cp -R` per conservare i collegamenti simbolici:

```
cp -R variants/aws-k8s-1.36-nvidia variants/aws-k8s-1.36-nvidia-595
cp -R sources/settings-defaults/aws-k8s-1.36-nvidia \
      sources/settings-defaults/aws-k8s-1.36-nvidia-595
```

Modifica`variants/aws-k8s-1.36-nvidia-595/Cargo.toml`:

```
- name = "aws-k8s-1_36-nvidia"
+ name = "aws-k8s-1_36-nvidia-595"

- "kmod-6.18-nvidia-r580-tesla",
+ "kmod-6.18-nvidia-r595-tesla",
```

Modifica`sources/settings-defaults/aws-k8s-1.36-nvidia-595/Cargo.toml`:

```
- name = "settings-defaults-aws-k8s-1_36-nvidia"
+ name = "settings-defaults-aws-k8s-1_36-nvidia-595"
```

Fase 7: Registrare la variante

Registra la nuova variante in cinque file:

1. **`Cargo.toml`**— aggiungi il membro dell'area di lavoro:

```
 "variants/aws-k8s-1.36-nvidia",
```

```
+    "variants/aws-k8s-1.36-nvidia-595",
 "variants/aws-k8s-1.36-nvidia-fips",
```

2. **`sources/Cargo.toml`**— aggiungi il membro settings-defaults:

```
 "settings-defaults/aws-k8s-1.36-nvidia",
+    "settings-defaults/aws-k8s-1.36-nvidia-595",
```

3. **`packages/settings-defaults/settings-defaults.spec`**— aggiungi un `%package` blocco, voci in entrambi i cicli di compilazione e una `%files` sezione:

```
%package aws-k8s-1.36-nvidia-595
Summary: Settings defaults for the aws-k8s 1.36 nvidia-595 variant
Requires: %{_cross_os}variant(aws-k8s-1.36-nvidia-595)
Provides: %{_cross_os}settings-defaults(any)
Provides: %{_cross_os}settings-defaults(aws-k8s-1.36-nvidia-595)
Conflicts: %{_cross_os}settings-defaults(any)

%description aws-k8s-1.36-nvidia-595
%{summary}.
```

Aggiungi a entrambi i `for defaults in` loop:

```
    aws-k8s-1.36-nvidia \
+   aws-k8s-1.36-nvidia-595 \
    metal-dev \
```

Aggiungi la `%files` sezione:

```
%files aws-k8s-1.36-nvidia-595
%{_cross_defaultsdir}/aws-k8s-1.36-nvidia-595.toml
%{_cross_tmpfilesdir}/storewolf-defaults-aws-k8s-1.36-nvidia-595.conf
```

4. **`packages/settings-plugins/settings-plugins.spec`**— aggiungi una `Provides:` riga sotto`%package aws-k8s-nvidia`:

```
 Provides: %{_cross_os}settings-plugin(aws-k8s-1.36-nvidia)
+Provides: %{_cross_os}settings-plugin(aws-k8s-1.36-nvidia-595)
 Conflicts: %{_cross_os}settings-plugin(any)
```

Fase 8: Aggiorna il file di blocco

L'aggiunta di un membro dell'area di lavoro invalida. `sources/Cargo.lock` Aggiorna:

```
cd ~/bottlerocket/sources
cargo update --workspace
```

> **Important**
>
> Non usare`cargo generate-lockfile`. Riscrive l'intero file di blocco ed elimina le dipendenze transitive, causando errori di versione duplicata. `cargo-deny`

Fase 9: Creare il file di licenza NVIDIA

NVIDIA limita la ridistribuzione delle fonti dei driver. È necessario aggiungere un riconoscimento esplicito della licenza prima di creare:

```
cat > ~/bottlerocket/Licenses.toml <<'EOF'
[nvidia]
spdx-id = "LicensesRef-NVIDIA-Customer-Use"
licenses = [
  { path = "LICENSE", license-url = "https://www.nvidia.com/en-us/drivers/nvidia-
license/" }
]
EOF
```

Fase 10: Creare e pubblicare l'AMI

Crea un file `Infra.toml` con le tue regioni target:

```
cat > ~/bottlerocket/Infra.toml <<'EOF'
[aws]
regions = ["us-west-2", "us-east-1", "us-east-2"]
EOF
```

Costruisci l'immagine:

```
cd ~/bottlerocket
```

```
cargo make \
  -e BUILDSYS_VARIANT=aws-k8s-1.36-nvidia-595 \
  -e BUILDSYS_UPSTREAM_SOURCE_FALLBACK=true \
  -e BUILDSYS_UPSTREAM_LICENSE_FETCH=true \
  -e BUILDSYS_JOBS=32
```

La prima build recupera l'SDK e le immagini del kit (~2 GB in totale). Le build successive richiedono 3-5 minuti su un host a 32 core.

Pubblica l'AMI:

```
cargo make \
  -e BUILDSYS_VARIANT=aws-k8s-1.36-nvidia-595 \
  -e PUBLISH_REGIONS=us-west-2,us-east-1,us-east-2 \
  ami
```

La build scrive gli ID AMI su`build/images/x86_64-aws-k8s-1.36-nvidia-595/latest/*-amis.json`. Usali nei tuoi gruppi di nodi gestiti da EKS, Karpenter o nei `EC2NodeClass` modelli di avvio.

Altre combinazioni che puoi creare

Questa procedura dettagliata sostituisce il ramo dei driver NVIDIA, ma puoi usare lo stesso approccio per qualsiasi pacchetto disponibile nei kit upstream. Gli esempi seguenti mostrano cosa è possibile assemblare senza forare un kit:

| Cosa vuoi | Cosa cambiare nella variante **`Cargo.toml`** |
|---|---|
| Diverso ramo dei driver NVIDIA | `kmod-6.18-nvidia-r580-tesla` → `kmod-6.18-nvidia-r595-tesla` |
| Versione del kernel diversa | `kernel-6.18` → `kernel-6.12` (aggiusta kmod di conseguenza) |
| Rimuovi completamente NVIDIA | Elimina le tre `nvidia-*` righe da `included-packages` |
| Aggiungi il supporto EFA | Aggiungi `kmod-6.18-efa` a `included-packages` |

| Cosa vuoi | Cosa cambiare nella variante **`Cargo.toml`** |
| --- | --- |
| Cambia la versione di runtime del contenitore | `containerd-2.2` → `containerd-2.1` |

Per informazioni sui pacchetti disponibili, consultate il kernel-kit sul GitHub sito Web e il core-kit sul sito Web. GitHub

> **Important**
>
> Il filesystem root di Bottlerocket è immutabile. Non è possibile installare pacchetti in fase di esecuzione. Ogni pacchetto dei kit è compilato in modo incrociato appositamente per Bottlerocket: gli RPM upstream standard non funzionano. Se hai bisogno di software che non sia già incluso in un kit, prendi in considerazione i contenitori bootstrap sul GitHub sito Web o i contenitori di hosting sul sito Web come alternativa di runtime. GitHub

Pulizia

Se non hai più bisogno dell'host di build, interrompi l'istanza EC2 per evitare addebiti continui. Le AMI persistono indipendentemente nel tuo account; annullale con la console EC2 o la CLI se non ti servono più.

Riepilogo

Questo argomento ha mostrato come creare una variante Bottlerocket personalizzata con un ramo di driver NVIDIA diverso. Il processo prevede la copia di una variante esistente, la modifica del riferimento di un pacchetto, la registrazione della nuova variante nell'area di lavoro e nelle specifiche RPM e l'esecuzione della build. Lo stesso approccio si applica a qualsiasi personalizzazione: scambio di versioni del kernel, aggiunta di pacchetti o creazione di varianti per nuove versioni di Kubernetes.

## Creare nodi con AMI ottimizzate per Ubuntu Linux

Canonical ha collaborato con Amazon EKS per creare AMI del nodo utilizzabili nei cluster.

Canonical offre un'immagine del sistema operativo del nodo Kubernetes integrata per lo scopo. Questa immagine Ubuntu ridotta è ottimizzata per Amazon EKS e include il kernel AWS personalizzato, sviluppato in collaborazione con AWS. Per ulteriori informazioni, consultare la pagina

Ubuntu on Amazon Elastic Kubernetes Service (EKS) e the section called "Ubuntu Linux". Per informazioni sul supporto, consultare la sezione Third-party software delle domande frequenti del Supporto AWS Premium.

## Crea nodi con Windows ottimizzato AMIs

Windows Amazon EKS AMIs optimized si basa su Windows Server 2019, Windows Server 2022 e Windows Server 2025. Sono configurate per fungere da immagine di base per i nodi Amazon EKS. Per impostazione predefinita, AMIs includono i seguenti componenti:

- kubelet
- kube-proxy
- AWS IAM Authenticator per Kubernetes
- csi-proxy
- containerd

> **Note**
>
> É possibile monitorare gli eventi di sicurezza o privacy per Windows Server con la Microsoft Security Update Guide.

Offerte AMIs Amazon EKS ottimizzate per contenitori Windows nelle seguenti varianti:

- AMI Windows Server, versione 2019 (Core) ottimizzata per Amazon EKS
- AMI Windows Server, versione 2019 (Full) ottimizzata per Amazon EKS
- AMI Core per Windows Server 2022 ottimizzata per Amazon EKS
- AMI completa per Windows Server 2022 ottimizzata per Amazon EKS
- AMI Core per Windows Server 2025 ottimizzata per Amazon EKS
- AMI completa per Windows Server 2025 ottimizzata per Amazon EKS

> **Important**
>
> - L'AMI Windows Server 20H2 (Core) ottimizzata per Amazon EKS è obsoleta. Non verranno rilasciate nuove versioni di questa AMI.

- Per garantire di disporre degli ultimi aggiornamenti di sicurezza AMIs per impostazione predefinita, Amazon EKS mantiene Windows ottimizzato negli ultimi 4 mesi. Ogni nuova AMI sarà disponibile per 4 mesi dal momento della pubblicazione iniziale. Dopo questo periodo, AMIs le versioni più vecchie vengono rese private e non sono più accessibili. Consigliamo di utilizzare le versioni più recenti AMIs per evitare vulnerabilità di sicurezza e perdere l'accesso alle versioni più vecchie AMIs che hanno raggiunto la fine del periodo di validità del supporto. Sebbene non possiamo garantire di poter fornire l'accesso a AMIs ciò che è stato reso privato, puoi richiedere l'accesso inviando un ticket a AWS Support.

## Calendario dei rilasci

Nella tabella seguente sono elencate le date di rilascio e di fine supporto delle versioni Windows per Amazon EKS. Se una data di fine del supporto è vuota, la versione è ancora supportata.

| Versione Windows | Rilascio Amazon EKS | Fine del supporto Amazon EKS |
|---|---|---|
| Windows Server 2025 Core | 27/01/2026 | |
| Windows Server 2025 completo | 27/01/2026 | |
| Windows Server 2022 Core | 17/10/2022 | |
| Windows Server 2022 Full | 17/10/2022 | |
| Windows Server 20H2 Core | 12/08/2021 | 09/08/2022 |
| Windows Server 2004 Core | 19/08/2020 | 14/12/2021 |
| Windows Server 2019 Core | 07/10/2019 | |
| Windows Server, versione 2019 (Full) | 07/10/2019 | |
| Windows Server, versione 1909 (Core) | 07/10/2019 | 08/12/2020 |

## Parametri di configurazione dello script di bootstrap

Quando crei un nodo Windows, sul nodo è presente uno script che consente la configurazione di diversi parametri. A seconda della configurazione, questo script può essere presente sul nodo in una posizione simile a: `C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`. È possibile specificare valori di parametri personalizzati specificandoli come argomenti dello script di bootstrap. Ad esempio, puoi aggiornare i dati utente nel modello di avvio. Per ulteriori informazioni, consulta the section called "Dati utente Amazon EC2".

Lo script include i seguenti parametri della riga di comando:

- `-EKSClusterName`: specifica il nome del cluster Amazon EKS a cui il nodo worker deve essere aggiunto.
- `-KubeletExtraArgs`: specifica argomenti aggiuntivi per `kubelet` (facoltativo).
- `-KubeProxyExtraArgs`: specifica argomenti aggiuntivi per `kube-proxy` (facoltativo).
- `-APIServerEndpoint`: specifica l'endpoint del server API del cluster Amazon EKS (facoltativo). Valido solo se usato con `-Base64ClusterCA`. Bypassa la chiamata a `Get-EKSCluster`.
- `-Base64ClusterCA`: specifica il contenuto CA del cluster codificato base64 (facoltativo). Valido solo se usato con `-APIServerEndpoint`. Bypassa la chiamata a `Get-EKSCluster`.
- `-DNSClusterIP`: sostituisce l'indirizzo IP da utilizzare per le query DNS all'interno del cluster (facoltativo). L'impostazione predefinita è `10.100.0.10` o `172.20.0.10` in base all'indirizzo IP dell'interfaccia primaria.
- `-ServiceCIDR`: sostituisce l'intervallo di indirizzi IP del servizio Kubernetes da cui vengono indirizzati i servizi del cluster. L'impostazione predefinita è `172.20.0.0/16` o `10.100.0.0/16` in base all'indirizzo IP dell'interfaccia primaria.
- `-ExcludedSnatCIDRs`— Un elenco di cose da escludere `IPv4` CIDRs da Source Network Address Translation (SNAT). Ciò significa che l'IP privato del pod indirizzabile nel VPC non verrebbe tradotto nell'indirizzo IP dell'indirizzo `IPv4` primario dell'interfaccia di rete elastica (ENI) dell'istanza per il traffico in uscita. Per impostazione predefinita, viene aggiunto il CIDR `IPv4` del VPC per il nodo Windows di Amazon EKS. La specificazione CIDRs di questo parametro esclude inoltre quanto specificato. CIDRs Per ulteriori informazioni, consulta the section called "Traffico in uscita".

Oltre ai parametri della riga di comando, è anche possibile specificare alcuni parametri delle variabili di ambiente. Quando si specifica un parametro della riga di comando, questo ha la precedenza

sulla rispettiva variabile di ambiente. Le variabili di ambiente devono essere definite come ambito di computer (o sistema) perché lo script bootstrap leggerà solo le variabili con ambito di computer.

Lo script tiene conto delle seguenti variabili di ambiente:

- `SERVICE_IPV4_CIDR`: per la definizione, fare riferimento al parametro della riga di comando `ServiceCIDR`.
- `EXCLUDED_SNAT_CIDRS`: dovrebbe essere una stringa separata da virgole. Fare riferimento al parametro della riga di comando `ExcludedSnatCIDRs` per la definizione.

Supporto dell'autenticazione gMSA

I pod Windows di Amazon EKS consentono diversi tipi di autenticazione dell'account del servizio gestito del gruppo (gMSA).

- Amazon EKS supporta le identità di dominio di Active Directory per l'autenticazione. Per ulteriori informazioni sulla gMSA aggiunta a un dominio, consulta la sezione Autenticazione di Windows su Amazon EKS Windowspods sul blog. AWS
- Amazon EKS offre un plug-in che consente ai nodi non-domain-joined Windows di recuperare le credenziali gMSA con un'identità utente portatile. Per ulteriori informazioni su gMSA senza dominio, consulta Domainless Windows Authentication for Amazon EKS Windowspods sul blog. AWS

## Immagini di container memorizzate nella cache

Amazon EKS Windows Optimized AMIs dispone di alcune immagini di container memorizzate nella cache per il `containerd` runtime. Le immagini dei container vengono memorizzate nella cache durante la creazione personalizzata AMIs utilizzando componenti di compilazione gestiti da Amazon. Per ulteriori informazioni, consulta the section called "Utilizzo del componente di compilazione gestito da Amazon".

Le seguenti immagini di container memorizzate nella cache sono per il runtime `containerd`:

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`
- `mcr.microsoft.com/windows/servercore`

## Ulteriori informazioni

Per ulteriori informazioni sull'uso di Windows ottimizzato per Amazon EKS AMIs, consulta le seguenti sezioni:

- Per informazioni dettagliate sull'esecuzione di carichi di lavoro su Windows accelerato ottimizzato per Amazon EKS AMIs, consulta. the section called "Configura le AMI GPU Windows"
- Per utilizzare Windows con gruppi di nodi gestiti, consulta the section called "Gruppi di nodi gestiti".
- Per avviare nodi Windows autogestiti, consulta the section called "Windows".
- Per informazioni sulla versione, consulta the section called "Ottenere informazioni sulla versione".
- Per recuperare la versione più recente IDs di Windows ottimizzata per Amazon EKS AMIs, consultathe section called "Scarica le ultime IDs".
- Per utilizzare Amazon EC2 Image Builder per creare finestre personalizzate ottimizzate AMIs per Amazon EKS, consulta. the section called "Compilazioni personalizzate"
- Per le best practice, consulta Amazon EKS optimized Windows AMI management nella Guida alle best practice di EKS.

## `Crea nodi Windows Server 2022 autogestiti con eksctl`

È possibile utilizzare il seguente `test-windows-2022.yaml` come riferimento per creare nodi Windows Server 2022 autogestiti. Sostituisci ogni *`example value`* con i valori in tuo possesso.

> **Note**
>
> Per eseguire nodi Windows Server 2022 autogestiti, è necessario utilizzare versione `eksctl` 0.116.0 o una versione successiva.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.36'

nodeGroups:
```

```
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

È quindi possibile creare i gruppi di nodi utilizzando il comando seguente.

```
eksctl create cluster -f test-windows-2022.yaml
```

## Recupero delle informazioni sulla versione delle AMI Windows

In questo argomento sono elencate le versioni delle AMI Windows ottimizzate per Amazon EKS e le relative versioni di kubelet, containerd e csi-proxy.

I metadati di un'AMI ottimizzata per Amazon EKS, incluso l'ID AMI per ogni variante, possono essere recuperati a livello di programmazione. Per ulteriori informazioni, consulta the section called "Scarica le ultime IDs".

Le versioni dell'AMI si basano sulla versione di Kubernetes e la data di rilascio dell'AMI è nel seguente formato:

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> I gruppi di nodi gestiti da Amazon EKS supportano i rilasci di novembre 2022 e successivi delle AMI Windows.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/nodes/eks-ami-versions-windows.adoc.atom
```

AMI di base per Windows Server 2025 ottimizzata per Amazon EKS

Le tabelle seguenti elencano le versioni attuali e precedenti dell'AMI Core per Windows Server 2025 ottimizzata per Amazon EKS.

Example

Kubernetes version 1.36

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.36-2026.08.15 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.07.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.06.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.05.19 | 1.36.1 | 2.2.3 | 1.2.1 | |

Kubernetes version 1.35

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.08.15 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.07.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.06.18 | 1.35.6 | 2.1.7 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.05.19 | 1.35.5 | 2.1.7 | 1.2.1 | |
| 1.35-2026.04.18 | 1.35.3 | 2.1.6 | 1.2.1 | |
| 1.35-2026.02.16 | 1.35.0 | 2.1.6 | 1.2.1 | |
| 1.35-2026-01-22 | 1.35.0 | 2.1.6 | 1.2.1 | |

AMI completa ottimizzata per Windows Server 2025 per Amazon EKS

Le tabelle seguenti elencano le versioni attuali e precedenti dell'AMI completa ottimizzata per Windows Server 2025 per Amazon EKS.

Example

Kubernetes version 1.36

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.36-2026.08.15 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.07.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.06.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.05.19 | 1.36.1 | 2.2.3 | 1.2.1 | |

Kubernetes version 1.35

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.08.15 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.07.21 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.06.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.05.19 | 1.35.5 | 2.1.7 | 1.2.1 | |
| 1.35-2026.04.18 | 1.35.3 | 2.1.6 | 1.2.1 | |
| 1.35-2026.02.16 | 1.35.0 | 2.1.6 | 1.2.1 | |
| 1.35-2026-01-22 | 1.35.0 | 2.1.6 | 1.2.1 | |

AMI Windows Server 2022 Core ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2022 Core ottimizzata per Amazon EKS.

Example

Kubernetes version 1.36

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.36-2026.08.15 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.07.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.06.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.05.19 | 1.36.1 | 2.2.3 | 1.2.1 | |

Kubernetes version 1.35

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.08.15 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.07.21 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.06.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.05.19 | 1.35.5 | 2.1.7 | 1.2.1 | |
| 1.35-2026.04.18 | 1.35.3 | 2.1.6 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.02.16 | 1.35.0 | 2.1.6 | 1.2.1 | |
| 1.35-2026-01-22 | 1.35.0 | 2.1.6 | 1.2.1 | |

Kubernetes version 1.34

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2026.08.15 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.07.18 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.06.18 | 1.34.9 | 2.1.6 | 1.2.1 | |
| 1.34-2026.05.19 | 1.34.8 | 2.1.6 | 1.2.1 | |
| 1.34-2026.04.18 | 1.34.6 | 2.1.6 | 1.2.1 | |
| 1.34-2026.02.13 | 1.34.3 | 2.1.6 | 1.2.1 | |
| 1.34-2026.01.22 | 1.34.2 | 2.1.6 | 1.2.1 | containerd è stato aggiornato a 2.1.6. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2025.12.15 | 1.34.2 | 2.1.4 | 1.2.1 | |
| 1.34-2025.11.14 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.10.18 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.09.13 | 1.34.0 | 2.1.4 | 1.2.1 | |

Kubernetes version 1.33

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.08.15 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.07.18 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.06.18 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.05.19 | 1.33.11 | 1.7.30 | 1.2.1 | |
| 1.33-2026.04.18 | 1.33.10 | 1.7.30 | 1.2.1 | |
| 1.33-2026.02.13 | 1.33.7 | 1.7.30 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.01.22 | 1.33.5 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.33-2025.12.15 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.11.14 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.10.18 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.09.13 | 1.33.4 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.33-2025.08.18 | 1.33.3 | 1.7.27 | 1.2.1 | |
| 1.33-2025.07.16 | 1.33.1 | 1.7.27 | 1.2.1 | |
| 1.33-2025.06.13 | 1.33.1 | 1.7.27 | 1.2.1 | |
| 1.33-2025.05.17 | 1.33.1 | 1.7.27 | 1.2.1 | |

Kubernetes version 1.32

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.32-2026.08.15 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.07.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.06.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.05.19 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.04.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.02.13 | 1.32.11 | 1.7.30 | 1.2.1 | |
| 1.32-2026.01.22 | 1.32.9 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.32-2025.12.15 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.11.14 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.10.18 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.09.13 | 1.32.8 | 1.7.28 | 1.2.1 | Modifica dei log del plugin |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| | | | | GMSA in Eventi di Windows |
| 1.32-2025.08.18 | 1.32.7 | 1.7.27 | 1.2.1 | |
| 1.32-2025.07.16 | 1.32.5 | 1.7.27 | 1.2.1 | |
| 1.32-2025.06.13 | 1.32.5 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.32-2025.05.17 | 1.32.5 | 1.7.20 | 1.2.1 | |
| 1.32-2025.04.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.03.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.02.18 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.01.15 | 1.32.0 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.31

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2026.08.15 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.07.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.06.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.05.19 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.04.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.02.13 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.01.22 | 1.31.13 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.31-2025.12.15 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.11.14 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.10.18 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.09.13 | 1.31.12 | 1.7.28 | 1.2.1 | Modifica dei log del plugin |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| | | | | GMSA in Eventi di Windows |
| 1.31-2025.08.18 | 1.31.11 | 1.7.27 | 1.2.1 | |
| 1.31-2025.07.16 | 1.31.9 | 1.7.27 | 1.2.1 | |
| 1.31-2025.06.13 | 1.31.9 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.31-2025.05.17 | 1.31.9 | 1.7.20 | 1.2.1 | |
| 1.31-2025.04.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.03.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.02.15 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.01.15 | 1.31.4 | 1.7.20 | 1.1.3 | |
| 1.31-2025.01.01 | 1.31.4 | 1.7.20 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.31-2024.12.13 | 1.31.3 | 1.7.20 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2024.11.12 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.10.08 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.10.01 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.09.10 | 1.31.0 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.30

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2026.07.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.06.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.05.19 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.04.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.02.13 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.01.22 | 1.30.14 | 1.7.30 | 1.2.1 | containerd è stato |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| | | | | aggiornato a `1.7.30`. |
| `1.30-2025.12.15` | `1.30.14` | `1.7.28` | `1.2.1` | |
| `1.30-2025.11.21` | `1.30.14` | `1.7.28` | `1.2.1` | |
| `1.30-2025.10.18` | `1.30.14` | `1.7.28` | `1.2.1` | |
| `1.30-2025.09.13` | `1.30.14` | `1.7.28` | `1.2.1` | Modifica dei log del plugin GMSA in Eventi di Windows |
| `1.30-2025.08.18` | `1.30.14` | `1.7.27` | `1.2.1` | |
| `1.30-2025.07.16` | `1.30.13` | `1.7.27` | `1.2.1` | |
| `1.30-2025.06.13` | `1.30.13` | `1.7.27` | `1.2.1` | `containerd` è stato aggiornato a `1.7.27`. |
| `1.30-2025.05.17` | `1.30.13` | `1.7.20` | `1.2.1` | |
| `1.30-2025.04.14` | `1.30.9` | `1.7.20` | `1.1.3` | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2025.03.14 | 1.30.9 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.30-2025.02.15 | 1.30.9 | 1.7.14 | 1.1.3 | |
| 1.30-2025.01.15 | 1.30.8 | 1.7.14 | 1.1.3 | |
| 1.30-2025.01.01 | 1.30.8 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.30-2024.12.11 | 1.30.7 | 1.7.14 | 1.1.3 | |
| 1.30-2024.11.12 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.10.08 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.09.10 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.08.13 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.07.10 | 1.30.2 | 1.7.14 | 1.1.2 | Include patch per CVE-2024-5321 . |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2024.06.17 | 1.30.0 | 1.7.14 | 1.1.2 | containerd è stato aggiornato a 1.7.14. |
| 1.30-2024.05.15 | 1.30.0 | 1.6.28 | 1.1.2 | |

Kubernetes version 1.29

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2026.02.13 | 1.29.15 | 1.7.30 | 1.2.1 | |
| 1.29-2026.01.22 | 1.29.15 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.29-2025.12.15 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.11.14 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.10.18 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.09.13 | 1.29.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2025.08.18 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.07.16 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.06.13 | 1.29.15 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.29-2025.05.17 | 1.29.15 | 1.7.20 | 1.2.1 | |
| 1.29-2025.04.14 | 1.29.13 | 1.7.20 | 1.1.3 | |
| 1.29-2025.03.14 | 1.29.13 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.29-2025.02.15 | 1.29.13 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.15 | 1.29.12 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.01 | 1.29.12 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.29-2024.12.11 | 1.29.10 | 1.7.14 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.11.12 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.10.08 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.09.10 | 1.29.6 | 1.7.14 | 1.1.3 | |
| 1.29-2024.08.13 | 1.29.6 | 1.7.14 | 1.1.3 | |
| 1.29-2024.07.10 | 1.29.6 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.29-2024.06.17 | 1.29.3 | 1.7.11 | 1.1.2 | |
| 1.29-2024.05.15 | 1.29.3 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.04.09 | 1.29.0 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |
| 1.29-2024.03.12 | 1.29.0 | 1.6.25 | 1.1.2 | |
| 1.29-2024.02.13 | 1.29.0 | 1.6.25 | 1.1.2 | |
| 1.29-2024.02.06 | 1.29.0 | 1.6.25 | 1.1.2 | Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo kubelet di rimozione di oggetti inutili (garbage collection). |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.01.11 | 1.29.0 | 1.6.18 | 1.1.2 | Escluso l'aggiornamento standalone di Windows KB5034439 per le AMI Windows Server 2022 Core. L'aggiornamento KB si applica solo alle installazioni Windows con una partizione e WinRE separata, che non sono incluse nelle nostre AMI Windows ottimizzate per Amazon EKS. |

Kubernetes version 1.28

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.11.14 | 1.28.15 | 1.7.28 | 1.2.1 | |
| 1.28-2025.10.18 | 1.28.15 | 1.7.28 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.09.13 | 1.28.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.28-2025.08.18 | 1.28.15 | 1.7.27 | 1.2.1 | |
| 1.28-2025.07.16 | 1.28.15 | 1.7.27 | 1.2.1 | |
| 1.28-2025.06.13 | 1.28.15 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.28-2025.05.17 | 1.28.15 | 1.7.20 | 1.2.1 | |
| 1.28-2025.04.14 | 1.28.15 | 1.7.20 | 1.1.3 | |
| 1.28-2025.03.14 | 1.28.15 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.28-2025.02.15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025.01.15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025-01-01 | 1.28.15 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2024.12.11 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2024.11.12 | 1.28.13 | 1.7.14 | 1.1.3 | |
| 1.28-2024.10.08 | 1.28.13 | 1.7.14 | 1.1.3 | |
| 1.28-2024.09.10 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.08.13 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.07.10 | 1.28.11 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.28-2024.06.17 | 1.28.8 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. |
| 1.28-2024.05.14 | 1.28.8 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2024.04.09 | 1.28.5 | 1.6.25 | 1.1.2 | containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |
| 1.28-2024.03.12 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2024.02.13 | 1.28.5 | 1.6.18 | 1.1.2 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2024.01.11 | 1.28.5 | 1.6.18 | 1.1.2 | Escluso l'aggiornamento standalone di Windows KB5034439 per le AMI Windows Server 2022 Core. L'aggiornamento KB si applica solo alle installazioni Windows con una partizione e WinRE separata, che non sono incluse nelle nostre AMI Windows ottimizzate per Amazon EKS. |
| 1.28-2023.12.12 | 1.28.3 | 1.6.18 | 1.1.2 | |
| 1.28-2023.11.14 | 1.28.3 | 1.6.18 | 1.1.2 | Include patch per CVE-2023-5528 . |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2023.10.19 | 1.28.2 | 1.6.18 | 1.1.2 | containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS ). |
| 1.28-2023-09.27 | 1.28.2 | 1.6.6 | 1.1.2 | Risolto un avviso di sicurezza in kubelet. |
| 1.28-2023.09.12 | 1.28.1 | 1.6.6 | 1.1.2 | |

AMI Windows Server 2022 Full ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2022 Full ottimizzata per Amazon EKS.

Example

Kubernetes version 1.36

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.36-2026.08.15 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.07.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.06.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.05.19 | 1.36.1 | 2.2.3 | 1.2.1 | |

Kubernetes version 1.35

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.08.15 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.07.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.06.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.05.19 | 1.35.5 | 2.1.7 | 1.2.1 | |
| 1.35-2026.04.18 | 1.35.3 | 2.1.6 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.02.16 | 1.35.0 | 2.1.6 | 1.2.1 | |
| 1.35-2026-01-22 | 1.35.0 | 2.1.6 | 1.2.1 | |

Kubernetes version 1.34

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2026.08.15 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.07.18 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.06.18 | 1.34.9 | 2.1.6 | 1.2.1 | |
| 1.34-2026.05.19 | 1.34.8 | 2.1.6 | 1.2.1 | |
| 1.34-2026.04.18 | 1.34.6 | 2.1.6 | 1.2.1 | |
| 1.34-2026.02.13 | 1.34.3 | 2.1.6 | 1.2.1 | |
| 1.34-2026.01.22 | 1.34.2 | 2.1.6 | 1.2.1 | containerd è stato aggiornato a 2.1.6. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2025.12.15 | 1.34.2 | 2.1.4 | 1.2.1 | |
| 1.34-2025.11.14 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.10.18 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.09.13 | 1.34.0 | 2.1.4 | 1.2.1 | |

Kubernetes version 1.33

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.08.15 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.07.21 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.06.18 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.05.19 | 1.33.11 | 1.7.30 | 1.2.1 | |
| 1.33-2026.04.18 | 1.33.10 | 1.7.30 | 1.2.1 | |
| 1.33-2026.02.13 | 1.33.7 | 1.7.30 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.01.22 | 1.33.5 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.33-2025.12.15 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.11.14 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.10.18 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.09.13 | 1.33.4 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.33-2025.08.18 | 1.33.3 | 1.7.27 | 1.2.1 | |
| 1.33-2025.07.16 | 1.33.1 | 1.7.27 | 1.2.1 | |
| 1.33-2025.06.13 | 1.33.1 | 1.7.27 | 1.2.1 | |
| 1.33-2025.05.17 | 1.33.1 | 1.7.27 | 1.2.1 | |

Kubernetes version 1.32

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.32-2026.08.15 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.07.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.06.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.05.19 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.04.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.02.13 | 1.32.11 | 1.7.30 | 1.2.1 | |
| 1.32-2026.01.22 | 1.32.9 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.32-2025.12.15 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.11.14 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.10.18 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.09.13 | 1.32.8 | 1.7.28 | 1.2.1 | Modifica dei log del plugin |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| | | | | GMSA in Eventi di Windows |
| 1.32-2025.08.18 | 1.32.7 | 1.7.27 | 1.2.1 | |
| 1.32-2025.07.16 | 1.32.5 | 1.7.27 | 1.2.1 | |
| 1.32-2025.06.13 | 1.32.5 | 1.7.27 | 1.2.1 | Aggiornamento di containerd a 1.7.27 |
| 1.32-2025.05.17 | 1.32.5 | 1.7.20 | 1.2.1 | |
| 1.32-2025.04.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.03.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.02.18 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.01.01 | 1.32.0 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.31

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2026.08.15 | 1.31.14 | 1.7.30 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2026.07.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.06.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.05.19 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.04.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.02.13 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.01.22 | 1.31.13 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.31-2025.12.15 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.11.14 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.10.18 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.09.13 | 1.31.12 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.31-2025.08.18 | 1.31.11 | 1.7.27 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2025.07.16 | 1.31.9 | 1.7.27 | 1.2.1 | |
| 1.31-2025.06.13 | 1.31.9 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.31-2025.05.17 | 1.31.9 | 1.7.20 | 1.2.1 | |
| 1.31-2025.04.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.03.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.02.15 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.01.15 | 1.31.4 | 1.7.20 | 1.1.3 | |
| 1.31-2025.01.01 | 1.31.4 | 1.7.20 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.31-2024.12.13 | 1.31.3 | 1.7.20 | 1.1.3 | |
| 1.31-2024.11.12 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.10.08 | 1.31.1 | 1.7.20 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2024.10.01 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.09.10 | 1.31.0 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.30

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2026.07.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.06.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.05.19 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.04.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.02.13 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.01.22 | 1.30.14 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.30-2025.12.15 | 1.30.14 | 1.7.28 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2025.11.21 | 1.30.14 | 1.7.28 | 1.2.1 | |
| 1.30-2025.10.18 | 1.30.14 | 1.7.28 | 1.2.1 | |
| 1.30-2025.09.13 | 1.30.14 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.30-2025.08.18 | 1.30.14 | 1.7.27 | 1.2.1 | |
| 1.30-2025.07.16 | 1.30.13 | 1.7.27 | 1.2.1 | |
| 1.30-2025.06.13 | 1.30.13 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.30-2025.05.17 | 1.30.13 | 1.7.20 | 1.2.1 | |
| 1.30-2025.04.14 | 1.30.9 | 1.7.20 | 1.1.3 | |
| 1.30-2025.03.14 | 1.30.9 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.30-2025.02.15 | 1.30.9 | 1.7.14 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2025.01.15 | 1.30.8 | 1.7.14 | 1.1.3 | |
| 1.30-2025.01.01 | 1.30.8 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.30-2024.12.11 | 1.30.7 | 1.7.14 | 1.1.3 | |
| 1.30-2024.11.12 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.10.08 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.09.10 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.08.13 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.07.10 | 1.30.2 | 1.7.14 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.30-2024.06.17 | 1.30.0 | 1.7.14 | 1.1.2 | containerd è stato aggiornato a 1.7.14. |
| 1.30-2024.05.15 | 1.30.0 | 1.6.28 | 1.1.2 | |

Kubernetes version 1.29

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2026.02.13 | 1.29.15 | 1.7.30 | 1.2.1 | |
| 1.29-2026.01.22 | 1.29.15 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.29-2025.12.15 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.11.14 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.10.18 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.09.13 | 1.29.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.29-2025.08.18 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.07.16 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.06.13 | 1.29.15 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2025.05.17 | 1.29.15 | 1.7.20 | 1.2.1 | |
| 1.29-2025.04.14 | 1.29.13 | 1.7.20 | 1.1.3 | |
| 1.29-2025.03.14 | 1.29.13 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.29-2025.02.15 | 1.29.13 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.15 | 1.29.12 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.01 | 1.29.12 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.29-2024.12.11 | 1.29.10 | 1.7.14 | 1.1.3 | |
| 1.29-2024.11.12 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.10.08 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.09.10 | 1.29.6 | 1.7.14 | 1.1.3 | |
| 1.29-2024.08.13 | 1.29.6 | 1.7.14 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.07.10 | 1.29.6 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.29-2024.06.17 | 1.29.3 | 1.7.11 | 1.1.2 | |
| 1.29-2024.05.15 | 1.29.3 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3. |
| 1.29-2024.04.09 | 1.29.0 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |
| 1.29-2024.03.12 | 1.29.0 | 1.6.25 | 1.1.2 | |
| 1.29-2024.02.13 | 1.29.0 | 1.6.25 | 1.1.2 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.02.06 | 1.29.0 | 1.6.25 | 1.1.2 | Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo kubelet di rimozione di oggetti inutili (garbage collection). |
| 1.29-2024.01.09 | 1.29.0 | 1.6.18 | 1.1.2 | |

Kubernetes version 1.28

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.11.14 | 1.28.15 | 1.7.28 | 1.2.1 | |
| 1.28-2025.10.18 | 1.28.15 | 1.7.28 | 1.2.1 | |
| 1.28-2025.09.13 | 1.28.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.28-2025.08.18 | 1.28.15 | 1.7.27 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.07.16 | 1.28.15 | 1.7.27 | 1.2.1 | |
| 1.28-2025.06.13 | 1.28.15 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.28-2025.05.17 | 1.28.15 | 1.7.20 | 1.2.1 | |
| 1.28-2025.04.14 | 1.28.15 | 1.7.20 | 1.1.3 | |
| 1.28-2025.03.14 | 1.28.15 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.28-2025.02.15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025.01.15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025.01.01 | 1.28.15 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.28-2024.12.11 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2024.11.12 | 1.28.13 | 1.7.14 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2024.10.08 | 1.28.13 | 1.7.14 | 1.1.3 | |
| 1.28-2024.09.10 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.08.13 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.07.10 | 1.28.11 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.28-2024.06.17 | 1.28.8 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. |
| 1.28-2024.05.14 | 1.28.8 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8. |
| 1.28-2024.04.09 | 1.28.5 | 1.6.25 | 1.1.2 | containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2024.03.12 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2024.02.13 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2024.01.09 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2023.12.12 | 1.28.3 | 1.6.18 | 1.1.2 | |
| 1.28-2023.11.14 | 1.28.3 | 1.6.18 | 1.1.2 | Include patch per CVE-2023-5528 . |
| 1.28-2023.10.19 | 1.28.2 | 1.6.18 | 1.1.2 | containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS ). |
| 1.28-2023-09.27 | 1.28.2 | 1.6.6 | 1.1.2 | Risolto un avviso di sicurezza in kubelet. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2023.09.12 | 1.28.1 | 1.6.6 | 1.1.2 | |

AMI Windows Server, versione 2019 (Core) ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2019 Core ottimizzata per Amazon EKS.

Example

Kubernetes version 1.36

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.36-2026.08.15 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.07.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.06.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.05.19 | 1.36.1 | 2.2.3 | 1.2.1 | |

Kubernetes version 1.35

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.08.15 | 1.35.6 | 2.1.7 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.07.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.06.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.05.19 | 1.35.5 | 2.1.7 | 1.2.1 | |
| 1.35-2026.04.18 | 1.35.3 | 2.1.6 | 1.2.1 | |
| 1.35-2026.02.16 | 1.35.0 | 2.1.6 | 1.2.1 | |
| 1.35-2026-01-22 | 1.35.0 | 2.1.6 | 1.2.1 | |

Kubernetes version 1.34

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2026.08.15 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.07.18 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.06.18 | 1.34.9 | 2.1.6 | 1.2.1 | |
| 1.34-2026.05.19 | 1.34.8 | 2.1.6 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2026.04.18 | 1.34.6 | 2.1.6 | 1.2.1 | |
| 1.34-2026.02.13 | 1.34.3 | 2.1.6 | 1.2.1 | |
| 1.34-2026.01.22 | 1.34.2 | 2.1.6 | 1.2.1 | containerd è stato aggiornato a 2.1.6. |
| 1.34-2025.12.15 | 1.34.2 | 2.1.4 | 1.2.1 | |
| 1.34-2025.11.14 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.10.18 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.09.13 | 1.34.0 | 2.1.4 | 1.2.1 | |

Kubernetes version 1.33

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.08.15 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.07.18 | 1.33.13 | 1.7.30 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.06.18 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.05.19 | 1.33.11 | 1.7.30 | 1.2.1 | |
| 1.33-2026.04.18 | 1.33.10 | 1.7.30 | 1.2.1 | |
| 1.33-2026.02.13 | 1.33.7 | 1.7.30 | 1.2.1 | |
| 1.33-2026.01.22 | 1.33.5 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.33-2025.12.15 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.11.14 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.10.18 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.09.13 | 1.33.4 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.33-2025.08.18 | 1.33.3 | 1.7.27 | 1.2.1 | |
| 1.33-2025.07.16 | 1.33.1 | 1.7.27 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2025.06.13 | 1.33.1 | 1.7.27 | 1.2.1 | |
| 1.33-2025.05.17 | 1.33.1 | 1.7.27 | 1.2.1 | |

Kubernetes version 1.32

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.32-2026.08.15 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.07.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.06.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.05.19 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.04.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.02.13 | 1.32.11 | 1.7.30 | 1.2.1 | |
| 1.32-2026.01.22 | 1.32.9 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.32-2025.12.15 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.11.14 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.10.18 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.09.13 | 1.32.8 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.32-2025.08.18 | 1.32.7 | 1.7.27 | 1.2.1 | |
| 1.32-2025.07.16 | 1.32.5 | 1.7.27 | 1.2.1 | |
| 1.32-2025.06.13 | 1.32.5 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.32-2025.05.17 | 1.32.5 | 1.7.20 | 1.2.1 | |
| 1.32-2025.04.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.03.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.02.18 | 1.32.1 | 1.7.20 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.32-2025.01.15 | 1.32.4 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.31

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2026.08.15 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.07.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.06.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.05.19 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.04.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.02.13 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.01.22 | 1.31.13 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.31-2025.12.15 | 1.31.13 | 1.7.28 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2025.11.14 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.10.18 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.09.13 | 1.31.12 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.31-2025.08.18 | 1.31.11 | 1.7.27 | 1.2.1 | |
| 1.31-2025.07.16 | 1.31.9 | 1.7.27 | 1.2.1 | |
| 1.31-2025.06.13 | 1.31.9 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.31-2025.05.17 | 1.31.9 | 1.7.20 | 1.2.1 | |
| 1.31-2025.04.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.03.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.02.15 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.01.15 | 1.31.4 | 1.7.20 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2025.01.01 | 1.31.4 | 1.7.20 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.31-2024.12.13 | 1.31.3 | 1.7.20 | 1.1.3 | |
| 1.31-2024.11.12 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.10.08 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.10.01 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.09.10 | 1.31.0 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.30

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2026.07.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.06.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.05.19 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.04.18 | 1.30.14 | 1.7.30 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2026.02.13 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.01.22 | 1.30.14 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.30-2025.12.15 | 1.30.14 | 1.7.28 | 1.2.1 | |
| 1.30-2025.11.21 | 1.30.14 | 1.7.28 | 1.2.1 | |
| 1.30-2025.10.18 | 1.30.14 | 1.7.28 | 1.2.1 | |
| 1.30-2025.09.13 | 1.30.14 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.30-2025.08.18 | 1.30.14 | 1.7.27 | 1.2.1 | |
| 1.30-2025.07.16 | 1.30.13 | 1.7.27 | 1.2.1 | |
| 1.30-2025.06.13 | 1.30.13 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.30-2025.05.17 | 1.30.13 | 1.7.20 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2025.04.14 | 1.30.9 | 1.7.20 | 1.1.3 | |
| 1.30-2025.03.14 | 1.30.9 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.30-2025-02-15 | 1.30.9 | 1.7.14 | 1.1.3 | |
| 1.30-2025.01.15 | 1.30.8 | 1.7.14 | 1.1.3 | |
| 1.30-2025.01.01 | 1.30.8 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.30-2024.12.11 | 1.30.7 | 1.7.14 | 1.1.3 | |
| 1.30-2024.11.12 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.10.08 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.09.10 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.08.13 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.07.10 | 1.30.2 | 1.7.14 | 1.1.2 | Include patch per CVE-2024-5321 . |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2024.06.17 | 1.30.0 | 1.7.14 | 1.1.2 | containerd è stato aggiornato a 1.7.14. |
| 1.30-2024.05.15 | 1.30.0 | 1.6.28 | 1.1.2 | |

Kubernetes version 1.29

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2026.02.13 | 1.29.15 | 1.7.30 | 1.2.1 | |
| 1.29-2026.01.22 | 1.29.15 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.29-2025.12.15 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.11.14 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.10.18 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.09.13 | 1.29.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2025.08.18 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.07.16 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.06.13 | 1.29.15 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.29-2025.05.17 | 1.29.15 | 1.7.20 | 1.2.1 | |
| 1.29-2025.04.14 | 1.29.13 | 1.7.20 | 1.1.3 | |
| 1.29-2025.03.14 | 1.29.13 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.29-2025.02.15 | 1.29.13 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.15 | 1.29.12 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.01 | 1.29.12 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.29-2024.12.11 | 1.29.10 | 1.7.14 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.11.12 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.10.08 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.09.10 | 1.29.6 | 1.7.14 | 1.1.3 | |
| 1.29-2024.08.13 | 1.29.6 | 1.7.14 | 1.1.3 | |
| 1.29-2024.07.10 | 1.29.6 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.29-2024.06.17 | 1.29.3 | 1.7.11 | 1.1.2 | |
| 1.29-2024.05.15 | 1.29.3 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.04.09 | 1.29.0 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |
| 1.29-2024.03.13 | 1.29.0 | 1.6.25 | 1.1.2 | |
| 1.29-2024.02.13 | 1.29.0 | 1.6.25 | 1.1.2 | |
| 1.29-2024.02.06 | 1.29.0 | 1.6.25 | 1.1.2 | Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo kubelet di rimozione di oggetti inutili (garbage collection). |
| 1.29-2024.01.09 | 1.29.0 | 1.6.18 | 1.1.2 | |

Kubernetes version 1.28

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.11.14 | 1.28.15 | 1.7.28 | 1.2.1 | |
| 1.28-2025.10.18 | 1.28.15 | 1.7.28 | 1.2.1 | |
| 1.28-2025.09.13 | 1.28.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.28-2025.08.18 | 1.28.15 | 1.7.27 | 1.2.1 | |
| 1.28-2025.07.16 | 1.28.15 | 1.7.27 | 1.2.1 | |
| 1.28-2025.06.13 | 1.28.15 | 1.7.20 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.28-2025.05.17 | 1.28.15 | 1.7.20 | 1.2.1 | |
| 1.28-2025.04.14 | 1.28.15 | 1.7.20 | 1.1.3 | |
| 1.28-2025.03.14 | 1.28.15 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.02.15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025-01-15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025-01-01 | 1.28.15 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.28-2024.12.11 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2024.11.12 | 1.28.13 | 1.7.14 | 1.1.3 | |
| 1.28-2024.10.08 | 1.28.13 | 1.7.14 | 1.1.3 | |
| 1.28-2024.09.10 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.08.13 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.07.10 | 1.28.11 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.28-2024.06.17 | 1.28.8 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2024.05.14 | 1.28.8 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8. |
| 1.28-2024.04.09 | 1.28.5 | 1.6.25 | 1.1.2 | containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |
| 1.28-2024.03.13 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2024.02.13 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2024.01.09 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2023.12.12 | 1.28.3 | 1.6.18 | 1.1.2 | |
| 1.28-2023.11.14 | 1.28.3 | 1.6.18 | 1.1.2 | Include patch per CVE-2023-5528 . |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2023.10.19 | 1.28.2 | 1.6.18 | 1.1.2 | containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS ). |
| 1.28-2023-09.27 | 1.28.2 | 1.6.6 | 1.1.2 | Risolto un avviso di sicurezza in kubelet. |
| 1.28-2023.09.12 | 1.28.1 | 1.6.6 | 1.1.2 | |

AMI Windows Server, versione 2019 (Full) ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2019 Full ottimizzata per Amazon EKS.

Example

Kubernetes version 1.36

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.36-2026.08.15 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.07.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.06.18 | 1.36.2 | 2.2.3 | 1.2.1 | |
| 1.36-2026.05.19 | 1.36.1 | 2.2.3 | 1.2.1 | |

Kubernetes version 1.35

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.08.15 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.07.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.06.18 | 1.35.6 | 2.1.7 | 1.2.1 | |
| 1.35-2026.05.19 | 1.35.5 | 2.1.7 | 1.2.1 | |
| 1.35-2026.04.18 | 1.35.3 | 2.1.6 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.35-2026.02.16 | 1.35.0 | 2.1.6 | 1.2.1 | |
| 1.35-2026-01-22 | 1.35.0 | 2.1.6 | 1.2.1 | |

Kubernetes version 1.34

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2026.08.15 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.07.21 | 1.34.9 | 2.1.7 | 1.2.1 | |
| 1.34-2026.06.18 | 1.34.9 | 2.1.6 | 1.2.1 | |
| 1.34-2026.05.19 | 1.34.8 | 2.1.6 | 1.2.1 | |
| 1.34-2026.04.18 | 1.34.6 | 2.1.6 | 1.2.1 | |
| 1.34-2026.02.13 | 1.34.3 | 2.1.6 | 1.2.1 | |
| 1.34-2026.01.22 | 1.34.2 | 2.1.6 | 1.2.1 | containerd è stato aggiornato a 2.1.6. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.34-2025.12.15 | 1.34.2 | 2.1.4 | 1.2.1 | |
| 1.34-2025.11.14 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.10.18 | 1.34.1 | 2.1.4 | 1.2.1 | |
| 1.34-2025.09.13 | 1.34.0 | 2.1.4 | 1.2.1 | |

Kubernetes version 1.33

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.08.15 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.07.18 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.06.18 | 1.33.13 | 1.7.30 | 1.2.1 | |
| 1.33-2026.05.19 | 1.33.11 | 1.7.30 | 1.2.1 | |
| 1.33-2026.04.18 | 1.33.10 | 1.7.30 | 1.2.1 | |
| 1.33-2026.02.13 | 1.33.7 | 1.7.30 | 1.2.1 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.33-2026.01.22 | 1.33.5 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.33-2025.12.15 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.11.14 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.10.18 | 1.33.5 | 1.7.28 | 1.2.1 | |
| 1.33-2025.09.13 | 1.33.4 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.33-2025.08.18 | 1.33.3 | 1.7.27 | 1.2.1 | |
| 1.33-2025.07.16 | 1.33.1 | 1.7.27 | 1.2.1 | |
| 1.33-2025.06.13 | 1.33.1 | 1.7.27 | 1.2.1 | |
| 1.33-2025.05.17 | 1.33.1 | 1.7.27 | 1.2.1 | |

Kubernetes version 1.32

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.32-2026.08.15 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.07.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.06.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.05.19 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.04.18 | 1.32.13 | 1.7.30 | 1.2.1 | |
| 1.32-2026.02.13 | 1.32.11 | 1.7.30 | 1.2.1 | |
| 1.32-2026.01.22 | 1.32.9 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.32-2025.12.15 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.11.14 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.10.18 | 1.32.9 | 1.7.28 | 1.2.1 | |
| 1.32-2025.09.13 | 1.32.8 | 1.7.28 | 1.2.1 | Modifica dei log del plugin |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| | | | | GMSA in Eventi di Windows |
| 1.32-2025.08.18 | 1.32.7 | 1.7.27 | 1.2.1 | |
| 1.32-2025.07.16 | 1.32.5 | 1.7.27 | 1.2.1 | |
| 1.32-2025.06.13 | 1.32.5 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.32-2025.05.17 | 1.32.5 | 1.7.20 | 1.2.1 | |
| 1.32-2025.04.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.03.14 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.02.18 | 1.32.1 | 1.7.20 | 1.1.3 | |
| 1.32-2025.01.15 | 1.32.0 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.31

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2026.08.15 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.07.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.06.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.05.19 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.04.18 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.02.13 | 1.31.14 | 1.7.30 | 1.2.1 | |
| 1.31-2026.01.22 | 1.31.13 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.31-2025.12.15 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.11.14 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.10.18 | 1.31.13 | 1.7.28 | 1.2.1 | |
| 1.31-2025.09.13 | 1.31.12 | 1.7.28 | 1.2.1 | Modifica dei log del plugin |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| | | | | GMSA in Eventi di Windows |
| 1.31-2025.08.18 | 1.31.11 | 1.7.27 | 1.2.1 | |
| 1.31-2025.07.16 | 1.31.9 | 1.7.27 | 1.2.1 | |
| 1.31-2025.06.13 | 1.31.9 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.31-2025.05.17 | 1.31.9 | 1.7.20 | 1.2.1 | |
| 1.31-2025.04.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.03.14 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.02.15 | 1.31.5 | 1.7.20 | 1.1.3 | |
| 1.31-2025.01.15 | 1.31.4 | 1.7.20 | 1.1.3 | |
| 1.31-2025.01.01 | 1.31.4 | 1.7.20 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.31-2024.12.13 | 1.31.3 | 1.7.20 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.31-2024.11.12 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.10.08 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.10.01 | 1.31.1 | 1.7.20 | 1.1.3 | |
| 1.31-2024.09.10 | 1.31.0 | 1.7.20 | 1.1.3 | |

Kubernetes version 1.30

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2026.07.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.06.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.05.19 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.04.18 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.02.13 | 1.30.14 | 1.7.30 | 1.2.1 | |
| 1.30-2026.01.22 | 1.30.14 | 1.7.30 | 1.2.1 | containerd è stato |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| | | | | aggiornato a `1.7.30`. |
| `1.30-2025.12.15` | `1.30.14` | `1.7.28` | `1.2.1` | |
| `1.30-2025.11.21` | `1.30.14` | `1.7.28` | `1.2.1` | |
| `1.30-2025.10.18` | `1.30.14` | `1.7.28` | `1.2.1` | |
| `1.30-2025.09.13` | `1.30.14` | `1.7.28` | `1.2.1` | Modifica dei log del plugin GMSA in Eventi di Windows |
| `1.30-2025.08.18` | `1.30.14` | `1.7.27` | `1.2.1` | |
| `1.30-2025.07.16` | `1.30.13` | `1.7.27` | `1.2.1` | |
| `1.30-2025.06.13` | `1.30.13` | `1.7.27` | `1.2.1` | `containerd` è stato aggiornato a `1.7.27`. |
| `1.30-2025.05.17` | `1.30.13` | `1.7.20` | `1.2.1` | |
| `1.30-2025.04.14` | `1.30.9` | `1.7.20` | `1.1.3` | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2025.03.14 | 1.30.9 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.30-2025.02.15 | 1.30.9 | 1.7.14 | 1.1.3 | |
| 1.30-2025.01.15 | 1.30.8 | 1.7.14 | 1.1.3 | |
| 1.30-2025.01.01 | 1.30.8 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.30-2024.12.11 | 1.30.7 | 1.7.14 | 1.1.3 | |
| 1.30-2024.11.12 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.10.08 | 1.30.4 | 1.7.14 | 1.1.3 | |
| 1.30-2024.09.10 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.08.13 | 1.30.2 | 1.7.14 | 1.1.3 | |
| 1.30-2024.07.10 | 1.30.2 | 1.7.14 | 1.1.2 | Include patch per CVE-2024-5321 . |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.30-2024.06.17 | 1.30.0 | 1.7.14 | 1.1.2 | containerd è stato aggiornato a 1.7.14. |
| 1.30-2024.05.15 | 1.30.0 | 1.6.28 | 1.1.2 | |

Kubernetes version 1.29

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2026.02.13 | 1.29.15 | 1.7.30 | 1.2.1 | |
| 1.29-2026.01.22 | 1.29.15 | 1.7.30 | 1.2.1 | containerd è stato aggiornato a 1.7.30. |
| 1.29-2025.12.15 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.11.14 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.10.18 | 1.29.15 | 1.7.28 | 1.2.1 | |
| 1.29-2025.09.13 | 1.29.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2025.08.18 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.07.16 | 1.29.15 | 1.7.27 | 1.2.1 | |
| 1.29-2025.06.13 | 1.29.15 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.29-2025.05.17 | 1.29.15 | 1.7.20 | 1.2.1 | |
| 1.29-2025.04.14 | 1.29.13 | 1.7.20 | 1.1.3 | |
| 1.29-2025.03.14 | 1.29.13 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |
| 1.29-2025.02.15 | 1.29.13 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.15 | 1.29.12 | 1.7.14 | 1.1.3 | |
| 1.29-2025.01.01 | 1.29.12 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.29-2024.12.11 | 1.29.10 | 1.7.14 | 1.1.3 | |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.11.12 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.10.08 | 1.29.8 | 1.7.14 | 1.1.3 | |
| 1.29-2024.09.10 | 1.29.6 | 1.7.14 | 1.1.3 | |
| 1.29-2024.08.13 | 1.29.6 | 1.7.14 | 1.1.3 | |
| 1.29-2024.07.10 | 1.29.6 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.29-2024.06.17 | 1.29.3 | 1.7.11 | 1.1.2 | |
| 1.29-2024.05.15 | 1.29.3 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.29-2024.04.09 | 1.29.0 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |
| 1.29-2024.03.13 | 1.29.0 | 1.6.25 | 1.1.2 | |
| 1.29-2024.02.13 | 1.29.0 | 1.6.25 | 1.1.2 | |
| 1.29-2024.02.06 | 1.29.0 | 1.6.25 | 1.1.2 | Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo kubelet di rimozione di oggetti inutili (garbage collection). |
| 1.29-2024.01.09 | 1.29.0 | 1.6.18 | 1.1.2 | |

Kubernetes version 1.28

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.11.14 | 1.28.15 | 1.7.28 | 1.2.1 | |
| 1.28-2025.10.18 | 1.28.15 | 1.7.28 | 1.2.1 | |
| 1.28-2025.09.13 | 1.28.15 | 1.7.28 | 1.2.1 | Modifica dei log del plugin GMSA in Eventi di Windows |
| 1.28-2025.08.18 | 1.28.15 | 1.7.27 | 1.2.1 | |
| 1.28-2025.07.16 | 1.28.15 | 1.7.27 | 1.2.1 | |
| 1.28-2025.06.13 | 1.28.15 | 1.7.27 | 1.2.1 | containerd è stato aggiornato a 1.7.27. |
| 1.28-2025.05.17 | 1.28.15 | 1.7.20 | 1.2.1 | |
| 1.28-2025.04.14 | 1.28.15 | 1.7.20 | 1.1.3 | |
| 1.28-2025.03.14 | 1.28.15 | 1.7.20 | 1.1.3 | containerd è stato aggiornato a 1.7.20. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2025.02.15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025-01-15 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2025-01-01 | 1.28.15 | 1.7.14 | 1.1.3 | Include patch per CVE-2024-9042 . |
| 1.28-2024.12.11 | 1.28.15 | 1.7.14 | 1.1.3 | |
| 1.28-2024.11.12 | 1.28.13 | 1.7.14 | 1.1.3 | |
| 1.28-2024.10.08 | 1.28.13 | 1.7.14 | 1.1.3 | |
| 1.28-2024.09.10 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.08.13 | 1.28.11 | 1.7.14 | 1.1.3 | |
| 1.28-2024.07.10 | 1.28.11 | 1.7.11 | 1.1.2 | Include patch per CVE-2024-5321 . |
| 1.28-2024.06.17 | 1.28.8 | 1.7.11 | 1.1.2 | containerd è stato aggiornato a 1.7.11. |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2024.05.14 | 1.28.8 | 1.6.28 | 1.1.2 | containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8. |
| 1.28-2024.04.09 | 1.28.5 | 1.6.25 | 1.1.2 | containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di golang 1.22.1. |
| 1.28-2024.03.13 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2024.02.13 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2024.01.09 | 1.28.5 | 1.6.18 | 1.1.2 | |
| 1.28-2023.12.12 | 1.28.3 | 1.6.18 | 1.1.2 | |
| 1.28-2023.11.14 | 1.28.3 | 1.6.18 | 1.1.2 | Include patch per CVE-2023-5528 . |

| Versione AMI | Versione kubelet | Versione containerd | Versione csi-proxy | Note di rilascio |
|---|---|---|---|---|
| 1.28-2023.10.19 | 1.28.2 | 1.6.18 | 1.1.2 | containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS ). |
| 1.28-2023-09.27 | 1.28.2 | 1.6.6 | 1.1.2 | Risolto un avviso di sicurezza in kubelet. |
| 1.28-2023.09.12 | 1.28.1 | 1.6.6 | 1.1.2 | |

## Recupera l'AMI Microsoft Windows consigliata IDs

Durante la distribuzione dei nodi, è possibile specificare un ID per un'Amazon Machine Image (AMI) preconfigurata e ottimizzata per Amazon EKS. Per recuperare un ID AMI adatto alla configurazione desiderata, interroga l'API AWS Systems Manager Parameter Store. L'utilizzo di questa API elimina la necessità di cercare manualmente le AMI ottimizzate per Amazon EKS IDs. Per ulteriori informazioni, consulta GetParameter. Il principale IAM che usi deve disporre dell'autorizzazione IAM `ssm:GetParameter` per recuperare i metadati dell'AMI ottimizzata per Amazon EKS.

È possibile recuperare l'ID immagine della versione consigliata dell'AMI Windows ottimizzata per Amazon EKS più recente con il comando seguente che utilizza il parametro secondario `image_id`. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

- Sostituisci *`release`* con una delle seguenti opzioni.
  - Utilizzare *`2025`* per Windows Server 2025.
  - Utilizzare *`2022`* per Windows Server 2022.
  - Utilizzare *`2019`* per Windows Server 2019.
- Sostituisci *`installation-option`* con una delle seguenti opzioni. Per ulteriori informazioni, consulta Cos'è l'opzione di installazione Server Core in Windows Server.
  - Utilizzare *`Core`* per un'installazione minima con una superficie di attacco più piccola.
  - *`Full`*Da utilizzare per includere l'esperienza desktop di Windows.
- Sostituisci *`kubernetes-version`* con una versione della piattaforma supportata.
- Sostituisci *`region-code`* con una AWS regione supportata da Amazon EKS per la quale desideri l'ID AMI.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-release-
English-installation-option-EKS_Optimized-kubernetes-version/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Ecco un comando esemplificativo dopo la sostituzione dei segnaposto.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2022-
English-Core-EKS_Optimized-k8s-n-2/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Di seguito viene riportato un output di esempio.

```
ami-1234567890abcdef0
```

## Creazione di un'AMI Windows personalizzata con Image Builder

Puoi usare EC2 Image Builder per creare Windows personalizzati ottimizzati per Amazon EKS AMIs con una delle seguenti opzioni:

- Utilizzo di un'AMI Windows ottimizzata per Amazon EKS come base
- Utilizzo del componente di compilazione gestito da Amazon

In entrambi i casi, è necessario creare la propria ricetta di Image Builder. Per ulteriori informazioni, consulta la sezione Creazione di una nuova versione di una ricetta di immagine nella Guida per l'utente di Image Builder.

> **Important**
>
> I seguenti componenti gestiti da Amazon per `eks` includono patch per `CVE-2024-5321`.
>
> - `1.28.2` e versioni successive
> - `1.29.2` e versioni successive
> - `1.30.1` e versioni successive
> - Tutte le versioni per Kubernetes 1.31 e versioni successive

Utilizzo di un'AMI Windows ottimizzata per Amazon EKS come base

Questa opzione è il metodo consigliato per creare Windows AMIs personalizzati. Le finestre ottimizzate per Amazon EKS AMIs che forniamo vengono aggiornate più frequentemente rispetto al componente di build gestito da Amazon.

1. Inizia a creare la tua ricetta di Image Builder.
   a. Aprire la console EC2 Image Builder in /imagebuilder. https://console.aws.amazon.com
   b. Nel riquadro di navigazione di sinistra, scegli Ricette di immagini.
   c. Scegli Crea ricetta di immagine.
2. Nella sezione Dettagli della ricetta, inserisci un nome e una versione.
3. Specifica l'ID dell'AMI Windows ottimizzata per Amazon EKS nella sezione Immagine di base.
   a. Scegli Inserisci un ID dell'AMI personalizzata.
   b. Recupera l'ID dell'AMI per la versione del sistema operativo Windows richiesta. Per ulteriori informazioni, consulta the section called "Scarica le ultime IDs".
   c. Inserisci l'ID dell'AMI personalizzata. Se l'ID AMI non viene trovato, assicurati che la AWS regione per l'ID AMI corrisponda alla AWS regione mostrata nella parte superiore destra della console.

4. (Facoltativo) Per ottenere gli ultimi aggiornamenti di sicurezza, aggiungi il componente `update-windows` nella sezione Componenti di compilazione.
   a. Dall'elenco a discesa a destra della casella di ricerca Trova componenti per nome scegli Gestito da Amazon.
   b. Nella sezione Trova componenti per nome, inserisci `update-windows`.
   c. Seleziona la casella di controllo del risultato della ricerca **`update-windows`**. Il componente include le patch di Windows più recenti per il sistema operativo.
5. Completa gli input rimanenti della ricetta di immagine con le configurazioni richieste. Per ulteriori informazioni, consulta la sezione Creazione di una nuova versione di una ricetta di immagine (console) nella Guida per l'utente di Image Builder.
6. Scegli Crea ricetta.
7. Utilizza la nuova ricetta di immagine in una pipeline di immagini nuova o esistente. Una volta che la pipeline di immagini viene eseguita correttamente, l'AMI personalizzata verrà elencata come immagine di output e sarà pronta per l'uso. Per ulteriori informazioni, vedere Creare una pipeline di immagini utilizzando la procedura guidata della console EC2 Image Builder.

Utilizzo del componente di compilazione gestito da Amazon

Quando non è possibile utilizzare un'AMI Windows ottimizzata per Amazon EKS come base, puoi utilizzare il componente di compilazione gestito da Amazon. Questa opzione potrebbe essere in ritardo rispetto alle versioni di Kubernetes supportate più recenti.

1. Inizia a creare la tua ricetta di Image Builder.
   a. Aprire la console EC2 Image Builder in /imagebuilder. https://console.aws.amazon.com
   b. Nel riquadro di navigazione di sinistra, scegli Ricette di immagini.
   c. Scegli Crea ricetta di immagine.
2. Nella sezione Dettagli della ricetta, inserisci un nome e una versione.
3. Determina quale opzione utilizzerai per creare la tua AMI personalizzata nella sezione Immagine di base:
   - Seleziona immagini gestite: scegli Windows come Sistema operativo (OS) per le immagini. Quindi scegli una delle seguenti opzioni per Origine dell'immagine.
     - Quick start (gestito da Amazon): nell'elenco a discesa Nome immagine, seleziona una versione Windows Server supportata da Amazon EKS. Per ulteriori informazioni, consulta the section called "Windows".

- Immagini di mia proprietà: per Nome immagine, seleziona l'ARN della tua immagine con la tua licenza. L'immagine fornita non deve avere componenti Amazon EKS già installati.

- Inserisci l'ID dell'AMI personalizzata: per l'ID dell'AMI, inserisci l'ID della tua AMI con la tua licenza. L'immagine fornita non deve avere componenti Amazon EKS già installati.

4. Nella sezione Componenti di compilazione - Windows, esegui le seguenti operazioni:
   a. Dall'elenco a discesa a destra della casella di ricerca Trova componenti per nome scegli Gestito da Amazon.
   b. Nella casella di ricerca Trova componenti per nome, inserisci `eks`.
   c. Seleziona la casella di controllo accanto al risultato della ricerca **`eks-optimized-ami-windows`**, anche se il risultato restituito potrebbe non essere la versione desiderata.
   d. Nella casella di ricerca Trova componenti per nome, inserisci `update-windows`.
   e. Seleziona la casella di controllo accanto al risultato della ricerca update-windows. Il componente include le patch di Windows più recenti per il sistema operativo.
5. Nella sezione Componenti selezionati, esegui le seguenti operazioni:
   a. Scegli Opzioni di controllo delle versioni per **`eks-optimized-ami-windows`**.
   b. Scegli Specifica la versione del componente.
   c. Nel campo Versione del componente, inserisci, sostituendolo ***`version`*** con una versione di *`version.x`* Kubernetes supportata. L'immissione *`x`* di una parte del numero di versione indica di utilizzare la versione più recente del componente che è anche in linea con la parte della versione definita in modo esplicito. Presta attenzione all'output della console, in quanto ti avviserà se la versione desiderata è disponibile come componente gestito. Tieni presente che le versioni di Kubernetes più recenti potrebbero non essere disponibili per il componente di compilazione. Per ulteriori informazioni sulle versioni disponibili, consulta the section called "Recupero di informazioni sulle versioni dei componenti di `eks-optimized-ami-windows`".
6. Completa gli input rimanenti della ricetta di immagine con le configurazioni richieste. Per ulteriori informazioni, consulta la sezione Creazione di una nuova versione di una ricetta di immagine (console) nella Guida per l'utente di Image Builder.
7. Scegli Crea ricetta.
8. Utilizza la nuova ricetta di immagine in una pipeline di immagini nuova o esistente. Una volta che la pipeline di immagini viene eseguita correttamente, l'AMI personalizzata verrà elencata come immagine di output e sarà pronta per l'uso. Per ulteriori informazioni, vedere Creare una pipeline di immagini utilizzando la procedura guidata della console EC2 Image Builder.

Recupero di informazioni sulle versioni dei componenti di **`eks-optimized-ami-windows`**

È possibile recuperare informazioni specifiche relative a ciò che viene installato con ciascun componente. Ad esempio, puoi verificare quale versione di `kubelet` è installata. I componenti sono sottoposti a test funzionali sulle versioni del sistema operativo Windows supportate da Amazon EKS. Per ulteriori informazioni, consulta the section called "Calendario dei rilasci". Altre eventuali versioni del sistema operativo Windows non elencate come supportate o che hanno raggiunto la fine del supporto potrebbero non essere compatibili con il componente.

1. Aprire la console EC2 Image Builder in /imagebuilder. https://console.aws.amazon.com
2. Nel riquadro di navigazione a sinistra, scegli Componenti.
3. Dall'elenco a discesa a destra della casella di ricerca Trova componenti per nome, modifica la selezione Di mia proprietà con Quick start (gestito da Amazon).
4. Nella sezione Trova componenti per nome, inserire `eks`.
5. (Facoltativo) Se stai utilizzando una versione recente, ordina la colonna Versione in ordine decrescente selezionandola due volte.
6. Scegli il collegamento **`eks-optimized-ami-windows`** con la versione desiderata.

La Descrizione nella pagina risultante mostra le informazioni specifiche.

# Rileva i problemi di integrità dei nodi e abilita la riparazione automatica dei nodi

Lo stato del nodo si riferisce allo stato operativo e alla capacità di un nodo Kubernetes di eseguire efficacemente i carichi di lavoro. Un nodo integro mantiene la connettività di rete prevista, dispone di risorse di elaborazione e archiviazione sufficienti e può eseguire con successo i carichi di lavoro senza interruzioni.

Per aiutare a mantenere i nodi integri nei cluster EKS, EKS offre l'agente di monitoraggio dei nodi e la riparazione automatica dei nodi. Queste funzionalità vengono abilitate automaticamente con EKS Auto Mode Compute. È inoltre possibile utilizzare la riparazione automatica dei nodi con i gruppi di nodi gestiti da EKS e Karpenter e utilizzare l'agente di monitoraggio dei nodi EKS con qualsiasi tipo di elaborazione EKS ad eccezione di Fargate. AWS L'agente di monitoraggio dei nodi EKS e la riparazione automatica dei nodi sono più efficaci se usati insieme, ma possono anche essere usati singolarmente nei cluster EKS.

> **Important**
>
> L'agente di monitoraggio dei nodi e la riparazione automatica dei nodi sono disponibili solo su Linux. Queste funzionalità non sono disponibili su Windows.

## Agente di monitoraggio del nodo

L'agente di monitoraggio dei nodi EKS legge i log dei nodi per rilevare problemi di salute. Analizza i log per rilevare i guasti e fornisce informazioni sullo stato di salute dei nodi. Per ogni categoria di problemi rilevati, l'agente ne applica uno dedicato `NodeCondition` ai nodi di lavoro. Per informazioni dettagliate sui problemi di integrità dei nodi rilevati dall'agente di monitoraggio dei nodi EKS, vederethe section called "Rilevamento dello stato dei nodi".

Il calcolo in modalità automatica di EKS include l'agente di monitoraggio dei nodi. Per altri tipi di elaborazione EKS, puoi aggiungere l'agente di monitoraggio dei nodi come componente aggiuntivo EKS o gestirlo con strumenti Kubernetes come Helm. Per ulteriori informazioni, consulta the section called "Configura l'agente di monitoraggio del nodo".

Con l'agente di monitoraggio dei nodi EKS, le seguenti categorie di problemi di integrità dei nodi vengono evidenziate come condizioni del nodo. Nota, `ReadyDiskPressure`, e `MemoryPressure` sono condizioni standard dei nodi Kubernetes che vengono rilevate anche senza l'agente di monitoraggio dei nodi EKS.

| Condizione del nodo | Description |
| --- | --- |
| AcceleratedHardwareReady | AcceleratedHardwareReady indica se l'hardwar e accelerato (GPU, Neuron) sul nodo funziona correttamente. |
| ContainerRuntimeReady | ContainerRuntimeReady indica se il runtime del contenitore (containerd, ecc.) funziona correttamente ed è in grado di eseguire contenitori. |
| DiskPressure | DiskPressure è una condizione standard di Kubernetes che indica che il nodo sta subendo |

| Condizione del nodo | Description |
| --- | --- |
| | una pressione sul disco (spazio su disco insufficiente o I/O elevato). |
| KernelReady | KernelReady indica se il kernel funziona correttamente senza errori critici, attacchi di panico o esaurimento delle risorse. |
| MemoryPressure | MemoryPressure è una condizione standard di Kubernetes che indica che il nodo sta subendo una pressione della memoria (memoria disponibile insufficiente). |
| NetworkingReady | NetworkingReady indica se lo stack di rete del nodo funziona correttamente (interfacce, routing, connettività). |
| StorageReady | StorageReady indica se il sottosistema di archiviazione del nodo funziona correttamente (dischi, file system, I/O). |
| Pronto | Ready è la condizione standard di Kubernete s che indica che il nodo è integro e pronto ad accettare i pod. |

## Riparazione automatica dei nodi

La riparazione automatica dei nodi EKS monitora continuamente lo stato dei nodi, reagisce ai problemi rilevati e sostituisce o riavvia i nodi quando possibile. Ciò migliora l'affidabilità del cluster con un intervento manuale minimo e aiuta a ridurre i tempi di inattività delle applicazioni.

Di per sé, la riparazione automatica dei nodi EKS reagisce alle `Ready` condizioni del kubelet, a tutti gli oggetti del nodo eliminati manualmente e alle istanze del gruppo di nodi gestite da EKS che non riescono a unirsi al cluster. Quando la riparazione automatica dei nodi EKS è abilitata con l'agente di monitoraggio dei nodi installato, la riparazione automatica dei nodi EKS reagisce a condizioni aggiuntive del nodo:`AcceleratedHardwareReady`,,, `ContainerRuntimeReady` e. `KernelReady` `NetworkingReady` `StorageReady`

La riparazione automatica dei nodi EKS non reagisce a Kubernetes `DiskPressure` standard o alle condizioni dei nodi. `MemoryPressure` `PIDPressure` Queste condizioni spesso indicano problemi relativi al comportamento dell'applicazione, alla configurazione del carico di lavoro o ai limiti delle risorse piuttosto che errori a livello di nodo, il che rende difficile determinare un'azione di riparazione predefinita appropriata. In questi scenari, i carichi di lavoro sono soggetti al comportamento di eliminazione della pressione dei nodi Kubernetes.

Per ulteriori informazioni sulla riparazione automatica dei nodi EKS, consulta. the section called "Riparazione automatica dei nodi"

# Rileva i problemi di integrità dei nodi con l'agente di monitoraggio dei nodi EKS

Questo argomento descrive in dettaglio i problemi di integrità dei nodi rilevati dall'agente di monitoraggio dei nodi EKS, come tali problemi vengono rilevati sotto forma di condizioni o eventi del nodo e come configurare l'agente di monitoraggio dei nodi.

L'agente di monitoraggio dei nodi EKS può essere utilizzato con o senza la riparazione automatica dei nodi EKS. Per ulteriori informazioni sulla riparazione automatica dei nodi EKS, vederethe section called "Riparazione automatica dei nodi".

Il codice sorgente dell'agente di monitoraggio dei nodi EKS è pubblicato GitHub nel repository aws/eks-node-monitoring-agent.

## Problemi di integrità dei nodi

Le tabelle seguenti descrivono i problemi di integrità dei nodi che possono essere rilevati dall'agente di monitoraggio dei nodi. Esistono due tipi di problemi:

- Condizione: un problema del terminale che richiede un'azione di riparazione come la sostituzione o il riavvio dell'istanza. Quando la riparazione automatica è abilitata, Amazon EKS eseguirà un'azione di riparazione, sostituendo o riavviando il nodo. Per ulteriori informazioni, consulta the section called "Condizioni dei nodi".
- Evento: un problema temporaneo o una configurazione non ottimale del nodo. Non sarà effettuata alcuna operazione di riparazione automatica. Per ulteriori informazioni, consulta the section called "Eventi dei nodi".

## AcceleratedHardware problemi di salute dei nodi

La condizione di monitoraggio riguarda `AcceleratedHardwareReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione". Gli eventi e le condizioni nella tabella seguente riguardano i problemi di salute dei nodi relativi a NVIDIA e Neuron.

| Nome | Gravità | Description | Azione di riparazione |
|---|---|---|---|
| DCGMDiagnosticFailure | Condizione | Un caso di test della suite di test della diagnostica attiva DCGM non è riuscito. | Nessuno |
| DCGMError | Condizione | La connessione al processo host DCGM è stata interrotta o non è stato possibile stabilirla. | Nessuno |
| DCGMFieldError[Codice] | Event | DCGM ha rilevato il degrado della GPU tramite un identificatore di campo. | Nessuno |
| DCGMHealthCode[Codice] | Event | Un controllo sanitario DCGM non è riuscito in modo non fatale. | Nessuno |
| DCGMHealthCode[Codice] | Condizione | Un controllo sanitario DCGM non è riuscito in modo fatale. | Nessuno |
| NeuronDMAError | Condizione | Un motore DMA ha rilevato un errore irreversibile. | Replace (Sostituisci) |
| NeuronHBMUncorrectableError | Condizione | Un HBM ha riscontrato un errore non | Replace (Sostituisci) |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| | | correggibile e ha prodotto risultati errati. | |
| NeuronNCUncorrectableError | Condizione | È stato rilevato un errore di memoria non correggibile di Neuron Core. | Replace (Sostituisci) |
| NeuronSRAMUncorrectableError | Condizione | Una SRAM su chip ha rilevato un errore di parità e ha prodotto risultati errati. | Replace (Sostituisci) |
| NvidiaDeviceCountMismatch | Event | Il numero di GPU visibili tramite NVML non è coerente con il numero di dispositivi NVIDIA sul file system. | Nessuno |
| NvidiaDoubleBitError | Condizione | Un errore a doppio bit è stato prodotto dal driver della GPU. | Replace (Sostituisci) |
| Errore NVIDIAN CCL | Event | Si è verificato un errore segfault nella libreria NVIDIA Collective Communications (). `libnccl` | Nessuno |
| NvidiaNVLinkError | Condizione | Gli errori NVLink sono stati segnalati dal driver della GPU. | Replace (Sostituisci) |

| Nome | Gravità | Description | Azione di riparazione |
|---|---|---|---|
| NvidiaPCIeError | Event | I replay PCIe sono stati attivati per il ripristino dagli errori di trasmissione. | Nessuno |
| NvidiaPageRetirement | Event | Il driver della GPU ha contrassegnato una pagina di memoria come ritirata. Ciò può verificarsi se si riscontra un singolo errore a doppio bit o due errori a singolo bit nello stesso indirizzo. | Nessuno |
| NvidiaPowerError | Event | L'utilizzo energetico delle GPU ha superato le soglie consentite. | Nessuno |
| NvidiaThermalError | Event | Lo stato termico delle GPU ha superato le soglie consentite. | Nessuno |
| Errore nvidiaXID [Code] | Condizione | Si è verificato un errore critico della GPU. | Sostituisci o riavvia |
| NvidiaXID[Code]Warning | Event | Si è verificato un errore non critico della GPU. | Nessuno |

## Codici di errore NVIDIA XID

L'agente di monitoraggio dei nodi rileva gli errori NVIDIA XID dai log del kernel della GPU. Gli errori XID si dividono in due categorie:

- Well-known Codici XID: errori critici che impostano una condizione del nodo (`AcceleratedHardwareReady=False`) e attivano la riparazione automatica quando abilitati. Il formato del codice motivo è`NvidiaXID[Code]Error`. I noti codici XID rilevati dall'agente di monitoraggio dei nodi EKS potrebbero non rappresentare l'elenco completo dei codici XID NVIDIA che richiedono azioni di riparazione.
- Codici XID sconosciuti: registrati solo come eventi Kubernetes. Questi non attivano la riparazione dell'auto. Il formato del codice motivo è`NvidiaXID[Code]Warning`. Per indagare sugli errori XID sconosciuti, controllate i log del kernel con. `dmesg | grep -i nvrm`

Per ulteriori informazioni sugli errori XID, consulta Errori Xid nella Documentazione di implementazione e gestione delle GPU NVIDIA. Per ulteriori informazioni sui singoli messaggi XID, consulta Comprensione dei messaggi Xid nella Documentazione di implementazione e gestione delle GPU NVIDIA.

La tabella seguente elenca i codici XID più diffusi, il loro significato e l'azione di riparazione dei nodi predefinita, se abilitata.

| Codice XID | Description | Azione di riparazione |
|---|---|---|
| 46 | Elaborazione interrotta della GPU: la GPU ha interrotto l'elaborazione a causa di un timeout interno e richiede un ripristino della GPU per il ripristino. | Riavvio |
| 48 | Errore ECC a doppio bit: si è verificato un errore a doppio bit non correggibile nella memoria della GPU, che indica un potenziale degrado dell'hardware. | Riavvio |
| 54 | Alimentazione ausiliaria non collegata: l'alimentazione ausiliaria non è collegata alla scheda GPU, il che indica in genere che i connettori di alimentazione non sono posizionati correttamente. | Riavviare |
| 62 | Arresto del microcontroller interno: il microcontroller interno della GPU si è arrestato, indicando un guasto del firmware o dell'hardware che richiede il ripristino della GPU. | Riavvio |

| Codice XID | Description | Azione di riparazione |
|---|---|---|
| 63 | Evento di rimappatura della memoria GPU: il driver GPU ha rimappato una parte della memoria GPU a causa di errori rilevati. Questo problema è spesso recuperabile. | Riavviare |
| 64 | Errore di rimappatura della memoria della GPU: la GPU non è riuscita a rimappare la memoria difettosa, il che indica problemi hardware. | Replace (Sostituisci) |
| 74 | Errore NVLink: si è verificato un errore nell'interconnessione NVLink ad alta velocità tra le GPU. | Replace (Sostituisci) |
| 79 | La GPU è caduta dal bus: la GPU non è più accessibile tramite PCIe, in genere indica un guasto hardware o un problema di alimentazione. | Replace (Sostituisci) |
| 95 | Errore di memoria non contenuta: si è verificato un errore di memoria che potrebbe aver influito su altre applicazioni o sulla memoria di sistema. | Riavvio |
| 109 | Timeout del cambio di contesto: un cambio di contesto della GPU non è stato completato in tempo, il che indica che la GPU è bloccata e richiede un ripristino. | Riavviare |
| 110 | Errore di sicurezza: è stato rilevato un errore di sicurezza sulla GPU, che richiede un ripristino della GPU e, se persiste, un'analisi dell'hardware. | Riavvio |
| 119 | Timeout GSP RPC: la comunicazione con il processore di sistema GPU è scaduta, probabilmente a causa di problemi del firmware. | Replace (Sostituisci) |
| 120 | Errore GSP: si è verificato un errore nel processore di sistema GPU. | Replace (Sostituisci) |

| Codice XID | Description | Azione di riparazione |
|---|---|---|
| 136 | Allenamento del collegamento non riuscito: la GPU non è riuscita a stabilire una connessione NVLink sana durante l'addestramento del collegamento. | Riavviare |
| 140 | Errore ECC non recuperato: un errore ECC è sfuggito al contenimento e potrebbe aver danneggiato i dati. | Riavvio |
| 142 | Errore NVENC3: l'hardware del codificatore video NVENC3 si è guasto e non è disponibile alcun ripristino automatico. Si applica a GB200. | Replace (Sostituisci) |
| 143 | Errore di inizializzazione della GPU: la GPU non è stata inizializ zata correttamente. | Riavvio |
| 151 | Errore di rotazione della chiave: la rotazione delle chiavi di calcolo riservato non è riuscita. Si applica a H100, B100 e GB200. | Replace (Sostituisci) |
| 155 | Errore definito dal software NVLink: su NVLink è stato segnalato un errore definito dal software. | Riavviare |
| 156 | Evento di ritiro delle risorse: la GPU ha ritirato una risorsa hardware a causa di errori rilevati. | Riavvio |
| 158 | Timeout irreversibile della GPU: si è verificato un timeout irreversibile sulla GPU. | Riavvio |

Per visualizzare le condizioni correnti del nodo relative allo stato della GPU, esegui il comando seguente.

```
kubectl get nodes -o custom-
columns='NAME:.metadata.name,ACCELERATOR_READY:.status.conditions[?
(@.type=="AcceleratedHardwareReady")].status,REASON:.status.conditions[?
(@.type=="AcceleratedHardwareReady")].reason'
```

Per visualizzare XID-related gli eventi sul tuo cluster, esegui uno dei seguenti comandi.

```
kubectl get events | grep -i "NvidiaXID"
```

## ContainerRuntime problemi di salute dei nodi

La condizione di monitoraggio riguarda `ContainerRuntimeReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

| Nome | Gravità | Description | Azione di riparazione |
|---|---|---|---|
| ContainerRuntimeFailed | Event | Il runtime del container non è riuscito a creare un container, probabilmente correlato a eventuali problemi segnalati se si verificano ripetutamente. | Nessuno |
| DeprecatedContainerdConfiguration | Event | Un'immagine del contenitore che utilizza il manifesto dell'immagine obsoleta versione 2, schema 1, è stata recentemente inserita nel nodo. `containerd` | Nessuno |
| KubeletFailed | Event | Il kubelet è entrato in uno stato di errore. | Nessuno |
| LivenessProbeFailures | Event | È stato rilevato un errore della sonda liveness, che potrebbe indicare problemi nel codice dell'applicazione o | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| | | nei valori di timeout insufficienti se si verificano ripetutam ente. | |
| PodStuckTerminating | Condizione | Un pod è o è rimasto bloccato nella terminazione per un periodo di tempo eccessivo, il che può essere causato da errori CRI che impediscono la progressione dello stato del pod. | Replace (Sostituisci) |
| ReadinessProbeFail ures | Event | È stato rilevato un errore della sonda di preparazione, che potrebbe indicare problemi nel codice dell'applicazione o nei valori di timeout insufficienti se si verificano ripetutam ente. | Nessuno |
| [Nome] RepeatedR estart | Event | Un'unità systemd si riavvia frequente mente. | Nessuno |
| ServiceFailedToStart | Event | Impossibile avviare un'unità systemd. | Nessuno |

## Problemi di integrità dei nodi del kernel

La condizione di monitoraggio riguarda `KernelReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

| Nome | Gravità | Description | Azione di riparazione |
|---|---|---|---|
| AppBlocked | Event | L'attività è stata bloccata per un lungo periodo di tempo dalla pianificazione, in genere a causa del blocco in ingresso o in uscita. | Nessuno |
| AppCrash | Event | Un'applicazione sul nodo si è bloccata. | Nessuno |
| ApproachingKernelPidMax | Event | Il numero di processi si sta avvicinando al numero massimo di PID disponibili per l'`kernel.pid_max` impostazione corrente, dopo di che non è possibile avviare altri processi. | Nessuno |
| ApproachingMaxOpenFiles | Event | Il numero di file aperti è simile al numero massimo di file aperti possibili date le impostazioni correnti del kernel, dopodiché l'apertura di nuovi file avrà esito negativo. | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| ConntrackExceededKernel | Event | Il rilevamento delle connessioni ha superato il valore massimo per il kernel e non è stato possibile stabilire nuove connessioni, il che può causare la perdita di pacchetti. | Nessuno |
| ExcessiveZombieProcesses | Event | I processi che non possono essere recuperati completamente si accumulano in gran numero, il che indica problemi di applicazione e può portare al raggiungimento dei limiti dei processi di sistema. | Nessuno |
| ForkFailedOutOfPIDs | Condizione | Una chiamata fork o exec non è riuscita per via dell'esaurimento degli ID di processo o della memoria del sistema, il che può essere causato da processi zombie o dall'esaurimento della memoria fisica. | Replace (Sostituisci) |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| KernelBug | Event | Un bug del kernel è stato rilevato e segnalato dal kernel Linux stesso, sebbene a volte ciò può essere causato da nodi con un elevato utilizzo della CPU o della memoria, con conseguente ritardo nell'elaborazione degli eventi. | Nessuno |
| LargeEnvironment | Event | Il numero di variabili di ambiente per questo processo è maggiore del previsto, potenzialmente causato da molti servizi `enableServiceLinks` impostati su true, il che può causare problemi di prestazioni. | Nessuno |
| RapidCron | Event | Un cron job è eseguito più velocemente di ogni cinque minuti su questo nodo, il che può influire sulle prestazioni se il processo consuma risorse significative. | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| SoftLockup | Event | La CPU si è bloccata per un periodo di tempo specificato. | Nessuno |

## Problemi di integrità dei nodi di rete

La condizione di monitoraggio riguarda `NetworkingReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| BandwidthInExceeded | Event | I pacchetti accodati o rilasciati perché la larghezza di banda aggregata in ingresso ha superato il valore massimo per l'istanza. | Nessuno |
| BandwidthOutExceeded | Event | I pacchetti sono stati accodati o rilasciati perché la larghezza di banda aggregata in uscita ha superato il valore massimo per l'istanza. | Nessuno |
| ConntrackExceeded | Event | Il rilevamento delle connessioni ha superato il valore massimo per l'istanza e non è stato possibile stabilire nuove connessioni, il che | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
|---|---|---|---|
| | | può determinare la perdita di pacchetti. | |
| EFAErrorMetric | Event | Le metriche dei driver EFA mostrano che esiste un'interfaccia con un peggioramento delle prestazioni. | Nessuno |
| IPAMDInconsistentState | Event | Lo stato del checkpoint IPAMD su disco non riflette gli IP nel runtime del contenitore. | Nessuno |
| IPAMDNoIPs | Event | IPAMD ha esaurito gli indirizzi IP. | Nessuno |
| IPAMDNotReady | Condizione | IPAMD non riesce a connettersi al server API. | Replace (Sostituisci) |
| IPAMDNotRunning | Condizione | Il processo CNI di Amazon VPC non è stato trovato in esecuzione. | Replace (Sostituisci) |
| IPAMDRepeatedlyRestart | Event | Si sono verificati più riavvii del servizio IPAMD. | Nessuno |
| InterfaceNotRunning | Condizione | Questa interfaccia sembra non essere in esecuzione o ci sono problemi di rete. | Replace (Sostituisci) |

| Nome | Gravità | Description | Azione di riparazione |
|---|---|---|---|
| InterfaceNotUp | Condizione | Questa interfaccia sembra non essere avviata o ci sono problemi di rete. | Replace (Sostituisci) |
| KubeProxyNotReady | Event | Kube-proxy non è riuscito a controllare o elencare le risorse. | Nessuno |
| LinkLocalExceeded | Event | I pacchetti sono stati accodati o rilasciat i perché il PPS del traffico verso i servizi proxy locali ha superato il valore massimo per l'interfa ccia di rete. | Nessuno |
| MACAddressPolicyMisconfigured | Event | La configurazione del link systemd-n etworkd ha un valore errato. `MACAddressPolicy` | Nessuno |
| MissingDefaultRoutes | Event | Mancano le regole di routing predefinite. | Nessuno |
| Routes IP mancanti | Event | Mancano percorsi per gli IP Pod. | Nessuno |
| Regole IP mancanti | Event | Mancano delle regole per gli IP dei Pod. | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| MissingLoopbackInterface | Condizione | L'interfaccia di loopback non è presente in questa istanza, il che causa l'interruzione dei servizi a seconda della connettività locale. | Replace (Sostituisci) |
| NetworkSysctl | Event | Le `sysctl` impostazioni di rete di questo nodo sono potenzialmente errate. | Nessuno |
| PPSExceeded | Event | I pacchetti sono stati accodati o rilasciati perché il PPS bidirezionale ha superato il valore massimo per l'istanza. | Nessuno |
| PortConflict | Event | Se un Pod utilizza HostPort, può scrivere `iptables` regole che sovrascrivono le porte già associate dell'host, impedendo potenzialmente l'accesso al server API. `kubelet` | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| UnexpectedRejectRule | Event | È stata rilevata una `DROP` regola `REJECT` o imprevista nel traffico `iptables` previsto che potrebbe bloccare. | Nessuno |

## Problemi di integrità dei nodi di storage

La condizione di monitoraggio riguarda `StorageReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| EBSInstanceIOPSExceeded | Event | È stato superato il numero massimo di IOPS per l'istanza. | Nessuno |
| EBSInstanceThroughputExceeded | Event | Il throughput massimo per l'istanza è stato superato. | Nessuno |
| EBSVolume IOPSExceeded | Event | È stato superato il numero massimo di IOPS per un determinato volume EBS. | Nessuno |
| EBSVolume ThroughputExceeded | Event | È stato superato il throughput massimo per un determinato volume Amazon EBS. | Nessuno |
| EtcHostsMountFailed | Event | Il montaggio del kubelet generato | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
| --- | --- | --- | --- |
| | | `/etc/hosts` non è riuscito a causa del rimontaggio dei dati utente durante il funzionamento. `/var/lib/kubelet/pods kubelet-container` | |
| IODelays | Event | Ritardo di ingresso o uscita rilevato in un processo, che potrebbe indicare un approvvigionamento input-output insufficiente, se eccessivo. | Nessuno |
| KubeletDiskUsageSlow | Event | Segnala `kubelet` un utilizzo lento del disco durante il tentativo di accesso al filesystem. Ciò potrebbe indicare problemi di input-output del disco o problemi di file system insufficienti. | Nessuno |

| Nome | Gravità | Description | Azione di riparazione |
|---|---|---|---|
| XFSSmallAverageClusterSize | Event | La dimensione media del cluster XFS è piccola, il che indica un'eccessiva frammentazione dello spazio libero. Ciò può impedire la creazione di file nonostante gli inode disponibili o lo spazio libero. | Nessuno |

## Configura l'agente di monitoraggio del nodo

L'agente di monitoraggio dei nodi EKS viene distribuito come DaemonSet. Quando lo si distribuisce come componente aggiuntivo EKS, è possibile personalizzare l'installazione con i seguenti valori di configurazione. Per le configurazioni predefinite, fate riferimento alla tabella Helm dell'agente di monitoraggio dei nodi EKS.

| Opzione di configurazione | Description |
|---|---|
| `monitoringAgent.resources.requests.cpu` | Richiesta di risorse CPU per l'agente di monitoraggio. |
| `monitoringAgent.resources.requests.memory` | Richiesta di risorse di memoria per l'agente di monitoraggio. |
| `monitoringAgent.resources.limits.cpu` | Limite di risorse CPU per l'agente di monitoraggio. |
| `monitoringAgent.resources.limits.memory` | Limite di risorse di memoria per l'agente di monitoraggio. |
| `monitoringAgent.tolerations` | Tolleranze per la pianificazione dell'agente di monitoraggio sui nodi contaminati. |

| Opzione di configurazione | Description |
| --- | --- |
| `monitoringAgent.additionalArgs` | Argomenti aggiuntivi della riga di comando da passare all'agente di monitoraggio. |

> **Note**
>
> È possibile configurare `hostname-override` e `verbosity` utilizzare i componenti aggiuntivi EKS o l'installazione di Helm. `monitoringAgent.additionalArgs` Al momento non è possibile personalizzare l'agente di monitoraggio del nodo `probe-address` (`8002`) o `metrics-address` (`8003`) tramite args aggiuntivi con componenti aggiuntivi EKS o l'installazione di Helm.

L'agente di monitoraggio del nodo include un componente server NVIDIA DCGM (Data Center GPU Manager) () per il monitoraggio delle GPU NVIDIA. `nv-hostengine` Questo componente funziona solo su nodi che sono tipi di istanze GPU NVIDIA, come mostrato nel grafico Helm dell'agente. `nodeAffinity` Non è possibile utilizzare un'installazione NVIDIA DCGM esistente con l'agente di monitoraggio dei nodi EKS. Se hai bisogno di questa funzionalità, fornisci un feedback sul GitHub numero #2763 della roadmap EKS.

Quando si implementa l'agente di monitoraggio dei nodi EKS come componente aggiuntivo EKS, è possibile personalizzare l'installazione NVIDIA DCGM con i seguenti valori di configurazione.

| Opzione di configurazione | Description |
| --- | --- |
| `dcgmAgent.resources.requests.cpu` | Richiesta di risorse CPU per l'agente DCGM. |
| `dcgmAgent.resources.requests.memory` | Richiesta di risorse di memoria per l'agente DCGM. |
| `dcgmAgent.resources.limits.cpu` | Limite di risorse CPU per l'agente DCGM. |
| `dcgmAgent.resources.limits.memory` | Limite di risorse di memoria per l'agente DCGM. |

| Opzione di configurazione | Description |
| --- | --- |
| `dcgmAgent.tolerations` | Tolleranze per la pianificazione dell'agente DCGM sui nodi contaminati. |

È possibile utilizzare i seguenti comandi AWS CLI per ottenere informazioni utili sulle versioni e sullo schema del componente aggiuntivo EKS dell'agente di monitoraggio dei nodi EKS.

Scarica l'ultima versione del componente aggiuntivo dell'agente per la tua versione di Kubernetes. Sostituiscila `1.35` con la tua versione di Kubernetes.

```
aws eks describe-addon-versions \
  --addon-name eks-node-monitoring-agent \
  --kubernetes-version 1.35 \
  --query='addons[].addonVersions[].addonVersion'
```

Scarica lo schema del componente aggiuntivo dell'agente supportato nei componenti aggiuntivi EKS. Sostituiscilo `v1.5.1-eksbuild.1` con la tua versione per agenti.

```
aws eks describe-addon-configuration \
  --addon-name eks-node-monitoring-agent \
  --addon-version v1.5.1-eksbuild.1
```

## Ripara automaticamente i nodi nei cluster EKS

Questo argomento descrive in dettaglio il comportamento di riparazione automatica dei nodi EKS e come configurarlo per soddisfare le vostre esigenze. La riparazione automatica dei nodi EKS è abilitata di default nella modalità automatica EKS e può essere utilizzata con i gruppi di nodi gestiti da EKS e Karpenter.

Le azioni di riparazione automatica dei nodi EKS predefinite sono riassunte nella tabella seguente e si applicano al comportamento di EKS Auto Mode, dei gruppi di nodi gestiti da EKS e Karpenter. Quando si utilizza EKS Auto Mode o Karpenter`Replace`, tutte le azioni di `AcceleratedHardwareReady` riparazione sono supportate e solo i gruppi di nodi gestiti da EKS sono `Reboot` supportate come azioni di riparazione.

Per un elenco dettagliato dei problemi di salute dei nodi rilevati dall'agente di monitoraggio dei nodi EKS e le relative azioni di riparazione dei nodi, vedere. the section called “Rilevamento dello stato dei nodi”

| Condizione del nodo | Description | Riparazione dopo | Azioni di riparazione |
|---|---|---|---|
| AcceleratedHardwareReady | AcceleratedHardwareReady indica se l'hardware accelerato (GPU, Neuron) sul nodo funziona correttamente. | 10 m | Sostituisci o riavvia |
| ContainerRuntimeReady | ContainerRuntimeReady indica se il runtime del contenitore (containerd, ecc.) funziona correttamente ed è in grado di eseguire contenitori. | 30 m | Replace (Sostituisci) |
| DiskPressure | DiskPressure è una condizione standard di Kubernetes che indica che il nodo sta subendo una pressione sul disco (spazio su disco insufficiente o I/O elevato). | N/D | Nessuno |
| KernelReady | KernelReady indica se il kernel funziona correttamente senza errori critici, attacchi di panico o esaurimento delle risorse. | 30 m | Replace (Sostituisci) |
| MemoryPressure | MemoryPressure è una condizione standard di Kubernete | N/D | Nessuno |

| Condizione del nodo | Description | Riparazione dopo | Azioni di riparazione |
|---|---|---|---|
| | s che indica che il nodo sta subendo una pressione della memoria (memoria disponibile insuffici ente). | | |
| NetworkingReady | NetworkingReady indica se lo stack di rete del nodo funziona correttam ente (interfacce, routing, connettività). | 30 m | Replace (Sostituisci) |
| StorageReady | StorageReady indica se il sottosistema di archiviazione del nodo funziona correttam ente (dischi, file system, I/O). | 30 m | Replace (Sostituisci) |
| Pronto | Ready è la condizione standard di Kubernete s che indica che il nodo è integro e pronto ad accettare i pod. | 30 m | Replace (Sostituisci) |

Le azioni di riparazione automatica dei nodi EKS sono disabilitate per impostazione predefinita nei seguenti scenari. Le azioni di riparazione dei nodi in corso continuano in ogni scenario. Scopri the section called "Configura la riparazione automatica dei nodi" come sovrascrivere queste impostazioni predefinite.

Gruppi di nodi gestiti da EKS

- Il gruppo di nodi ha più di cinque nodi e più del 20% dei nodi del gruppo di nodi non è integro.

- Lo spostamento zonale del cluster viene attivato tramite l'Application Recovery Controller (ARC).

EKS Auto Mode e Karpenter

- Oltre il 20% dei nodi presenti non NodePool sono sani.
- In modalità standalone NodeClaims, il 20% dei nodi del cluster non è integro.

## Configura la riparazione automatica dei nodi

La riparazione automatica dei nodi non può essere configurata quando si utilizza la modalità automatica EKS ed è sempre abilitata con le stesse impostazioni predefinite di Karpenter.

Karpenter

Per utilizzare la riparazione automatica dei nodi con Karpenter, abilita il feature gate. `NodeRepair=true` È possibile abilitare i feature gate tramite l'opzione `--feature-gates` CLI o la variabile di `FEATURE_GATES` ambiente nella distribuzione di Karpenter. Per ulteriori informazioni, consulta la documentazione di Karpenter.

Gruppi di nodi gestiti

È possibile abilitare la riparazione automatica dei nodi durante la creazione di nuovi gruppi di nodi gestiti EKS o aggiornando i gruppi di nodi gestiti EKS esistenti.

- Console Amazon EKS: seleziona la casella di controllo Abilita riparazione automatica del nodo per il gruppo di nodi gestito. Per ulteriori informazioni, consulta the section called "Crea".
- AWS CLI: aggiungi `--node-repair-config enabled=true` al comando `eks create-nodegroup`or `eks update-nodegroup-config`.
- eksctl — Configure`managedNodeGroups.nodeRepairConfig.enabled: true`, vedi l'esempio in eksctl. GitHub

Quando si utilizzano i gruppi di nodi gestiti da EKS, è possibile controllare il comportamento di riparazione automatica dei nodi con le seguenti impostazioni.

Per controllare quando la riparazione automatica del nodo smette di agire, imposta una soglia basata sul numero di nodi non integri nel gruppo di nodi. Imposta il conteggio assoluto o la percentuale, ma non entrambi.

| Impostazione | Description |
| --- | --- |
| `maxUnhealthyNodeThresholdCount` | Il numero assoluto di nodi non integri al di sopra dei quali si interrompe la riparazione automatica del nodo. Utilizzatelo per limitare l'ambito delle riparazioni. |
| `maxUnhealthyNodeThresholdPercentage` | La percentuale di nodi non integri al di sopra della quale si interrompe la riparazione automatica del nodo (0-100). |

Per controllare quanti nodi vengono riparati contemporaneamente, puoi configurare il parallelismo di riparazione. Come per la soglia dei nodi non integri, imposta il conteggio assoluto o la percentuale, ma non entrambi.

| Impostazione | Description |
| --- | --- |
| `maxParallelNodesRepairedCount` | Il numero massimo di nodi da riparare contemporaneamente. |
| `maxParallelNodesRepairedPercentage` | La percentuale massima di nodi non integri da riparare contemporaneamente (0-100). |

Con `nodeRepairConfigOverrides`, è possibile personalizzare il comportamento di riparazione per condizioni specifiche. Utilizzalo quando hai bisogno di diverse azioni di riparazione o tempi di attesa per diversi tipi di problemi.

Ogni override richiede tutti i seguenti campi:

| Campo | Description |
| --- | --- |
| `nodeMonitoringCondition` | Il tipo di condizione del nodo riportato dall'agente di monitoraggio del nodo. Ad esempio: `AcceleratedHardwareReady`, `NetworkingReady`, `StorageReady`, `KernelReady`. |

| Campo | Description |
| --- | --- |
| `nodeUnhealthyReason` | Il codice causale specifico della condizione non salutare. Ad esempio, `NvidiaXID31Error` , `IPAMDNotRunning` . |
| `minRepairWaitTimeMins` | Il tempo minimo, in minuti, in cui la condizione deve persistere prima che il nodo diventi idoneo alla riparazione. Usalo per evitare di riparare i nodi per problemi temporanei. |
| `repairAction` | L'azione da intraprendere quando le condizioni sono soddisfatte. Valori validi: `Replace` (terminare e sostituire il nodo), `Reboot` (riavviare il nodo) o `NoAction` (nessuna azione di riparazione). |

L'esempio AWS CLI seguente crea un gruppo di nodi con impostazioni di riparazione personalizzate.

```
aws eks create-nodegroup \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --node-role arn:aws:iam::111122223333:role/NodeRole \
  --subnets subnet-0123456789abcdef0 \
  --node-repair-config '{
    "enabled": true,
    "maxUnhealthyNodeThresholdPercentage": 10,
    "maxParallelNodesRepairedCount": 3,
    "nodeRepairConfigOverrides": [
      {
        "nodeMonitoringCondition": "AcceleratedHardwareReady",
        "nodeUnhealthyReason": "NvidiaXID64Error",
        "minRepairWaitTimeMins": 5,
        "repairAction": "Replace"
      },
      {
        "nodeMonitoringCondition": "AcceleratedHardwareReady",
        "nodeUnhealthyReason": "NvidiaXID31Error",
        "minRepairWaitTimeMins": 15,
        "repairAction": "NoAction"
      }
```

```
    ]
  }'
```

Questa configurazione esegue le seguenti operazioni:

- Abilita la riparazione automatica del nodo
- Interrompe le azioni di riparazione quando più del 10% dei nodi non è integro
- Ripara fino a 3 nodi alla volta
- Sostituisce gli errori XID 64 (errore di rimappatura della memoria GPU) per sostituire il nodo dopo 5 minuti. L'impostazione predefinita è il riavvio dopo 10 minuti.
- Sostituisce gli errori XID 31 (errore della pagina di memoria della GPU) per non intraprendere alcuna azione. L'impostazione predefinita è il riavvio dopo 10 minuti.

## Visualizzazione dello stato di integrità dei nodi

Questo argomento spiega gli strumenti e i metodi disponibili per monitorare lo stato di integrità dei nodi nei cluster Amazon EKS. Le informazioni riguardano le condizioni dei nodi, gli eventi e i casi di rilevamento che aiutano a identificare e diagnosticare problemi a livello di nodo. Utilizza i comandi e gli schemi descritti qui per ispezionare le risorse dello stato dei nodi, interpretare le condizioni dello stato e analizzare gli eventi dei nodi per la risoluzione dei problemi operativi.

Puoi ottenere alcune informazioni sullo stato dei nodi con i comandi Kubernetes per tutti i nodi. E se utilizzi l'agente di monitoraggio dei nodi tramite la modalità automatica di Amazon EKS o il componente aggiuntivo gestito Amazon EKS, otterrai una più ampia varietà di segnali di nodo per aiutarti a risolvere i problemi. Le descrizioni dei problemi di integrità rilevati dall'agente di monitoraggio dei nodi sono disponibili anche nella dashboard di osservabilità. Per ulteriori informazioni, consulta the section called "Rilevamento dello stato dei nodi".

### Condizioni dei nodi

Le condizioni dei nodi rappresentano problemi terminali che richiedono azioni di riparazione come la sostituzione o il riavvio dell'istanza.

Per ottenere le condizioni per tutti i nodi:

```
kubectl get nodes -o 'custom-
columns=NAME:.metadata.name,CONDITIONS:.status.conditions[*].type,STATUS:.status.conditions[*].
```

Per ottenere condizioni dettagliate per un nodo specifico

```
kubectl describe node node-name
```

Esempio di output delle condizioni di un nodo integro:

```
  - lastHeartbeatTime: "2024-11-21T19:07:40Z"
    lastTransitionTime: "2024-11-08T03:57:40Z"
    message: Monitoring for the Networking system is active
    reason: NetworkingIsReady
    status: "True"
    type: NetworkingReady
```

Esempio di condizione di un nodo non integro con un problema di rete:

```
  - lastHeartbeatTime: "2024-11-21T19:12:29Z"
    lastTransitionTime: "2024-11-08T17:04:17Z"
    message: IPAM-D has failed to connect to API Server which could be an issue with
      IPTable rules or any other network configuration.
    reason: IPAMDNotReady
    status: "False"
    type: NetworkingReady
```

## Eventi dei nodi

Gli eventi dei nodi indicano problemi temporanei o configurazioni non ottimali.

Per ottenere tutti gli eventi segnalati dall'agente di monitoraggio dei nodi

Quando l'agente di monitoraggio dei nodi è disponibile, puoi eseguire il comando seguente.

```
kubectl get events --field-selector=reportingComponent=eks-node-monitoring-agent
```

Output di esempio:

```
LAST SEEN   TYPE      REASON       OBJECT
 MESSAGE
4s          Warning   SoftLockup   node/ip-192-168-71-251.us-west-2.compute.internal
 CPU stuck for 23s
```

Per ottenere eventi per tutti i nodi

```
kubectl get events --field-selector involvedObject.kind=Node
```

Per ottenere eventi per un nodo specifico

```
kubectl get events --field-selector involvedObject.kind=Node,involvedObject.name=node-name
```

Per osservare gli eventi in tempo reale

```
kubectl get events -w --field-selector involvedObject.kind=Node
```

Esempio di output di un evento:

```
LAST SEEN   TYPE     REASON           OBJECT         MESSAGE
2m          Warning  MemoryPressure   Node/node-1    Node experiencing memory pressure
5m          Normal   NodeReady        Node/node-1    Node became ready
```

### Comandi per la risoluzione dei problemi comuni

```
# Get comprehensive node status
kubectl get node node-name -o yaml

# Watch node status changes
kubectl get nodes -w

# Get node metrics
kubectl top node
```

## Recuperare i log dei nodi per un nodo gestito usando kubectl e S3

Scopri come recuperare i log dei nodi per un nodo gestito di Amazon EKS che dispone dell'agente di monitoraggio dei nodi.

### Prerequisiti

Verifica di disporre di quanto segue:

- Un cluster Amazon EKS esistente con agente di monitoraggio dei nodi. Per ulteriori informazioni, consulta the section called "Salute e riparazione dei nodi".
- Lo strumento di riga di comando `kubectl` installato e configurato per comunicare con il cluster.
- La AWS CLI è stata installata e ha effettuato l'accesso con autorizzazioni sufficienti per creare bucket e oggetti S3.
- Una versione recente di Python 3 installata
- L' AWS SDK per Python 3, Boto 3, è installato.

### Fase 1: Creare di una destinazione bucket S3 (facoltativo)

Se non disponi già di un bucket S3 in cui archiviare i log, creane uno. Usa il seguente comando AWS CLI. Per impostazione predefinita, il bucket è l'elenco di controllo degli accessi `private`. Sostituisci *`bucket-name`* con il nome univoco del bucket scelto.

```
aws s3api create-bucket --bucket <bucket-name>
```

### Fase 2: Creare un URL S3 prefirmato per HTTP Put

Amazon EKS restituisce i log dei nodi eseguendo un'operazione HTTP PUT su un URL specificato. In questo tutorial, genereremo un URL PUT HTTP S3 prefirmato.

I log verranno restituiti come gzip tarball, con l'estensione `.tar.gz`.

> **Note**
>
> È necessario utilizzare l' AWS API o un SDK per creare l'URL di caricamento S3 prefirmato per consentire a EKS di caricare il file di registro. Non puoi creare un URL di caricamento S3 prefirmato utilizzando la CLI AWS .

1. Determina in che punto del bucket desideri archiviare i log. Ad esempio, puoi usare *`2024-11-12/logs1.tar.gz`* come chiave.
2. Salva il seguente codice Python sul file *`presign-upload.py`*. Sostituisci *`<bucket-name>`* e *`<key>`*. La chiave deve terminare con `.tar.gz`.

```
import boto3; print(boto3.client('s3').generate_presigned_url(
```

```
    ClientMethod='put_object',
    Params={'Bucket': '<bucket-name>', 'Key': '<key>'},
    ExpiresIn=1000
))
```

3. Esegui lo script con

```
python presign-upload.py
```

4. Annota l'output dell'URL. Usa questo valore nella fase successiva come *`http-put-destination`*.

Per ulteriori informazioni, consulta Generare un URL predefinito per caricare un file nella documentazione AWS Boto3 SDK per Python.

### Fase 3: Creare una risorsa NodeDiagnostic

Identifica il nome del nodo da cui si desidera raccogliere i log.

Crea un manifesto `NodeDiagnostic` che utilizzi il nome del nodo come nome della risorsa e fornire una destinazione URL PUT HTTP.

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
    name: <node-name>
spec:
    logCapture:
        destination: http-put-destination
```

Applica il file manifesto al cluster.

```
kubectl apply -f nodediagnostic.yaml
```

Puoi controllare lo stato della raccolta descrivendo la risorsa `NodeDiagnostic`:

- Uno stato di `Success` o `SuccessWithErrors` indica che l'attività è stata completata e i log sono stati caricati nella destinazione fornita (`SuccessWithErrors` indica che alcuni log potrebbero mancare).

- Se lo stato è Failure, conferma che l'URL di caricamento è ben formato e non è scaduto.

```
kubectl describe nodediagnostics.eks.amazonaws.com/<node-name>
```

### Fase 4: Scaricare log da S3

Attendi circa un minuto prima di tentare di scaricare i log. Quindi, usa la CLI di S3 per scaricare i log.

```
# Once NodeDiagnostic shows Success status, download the logs
aws s3 cp s3://<bucket-name>/key ./<path-to-node-logs>.tar.gz
```

### Fase 5: Pulizia della NodeDiagnostic risorsa

- Le risorse `NodeDiagnostic` non vengono eliminate automaticamente. Devi eliminarle in autonomia dopo aver ottenuto gli artefatti del log.

```
# Delete the NodeDiagnostic resource
kubectl delete nodediagnostics.eks.amazonaws.com/<node-name>
```

## NodeDiagnostic destinazione **`del nodo`**

A partire dalla versione `v1.6.1-eksbuild.1` di Node Monitoring Agent, è disponibile un'opzione su cui impostare la destinazione della raccolta dei log`node`. L'utilizzo di questa destinazione porterà alla raccolta e alla persistenza temporanea dei log sul nodo per la raccolta successiva. Oltre a questa funzionalità, all'interno del GitHub repository di Node Monitoring Agent è presente un `kubectl` plug-in che è possibile installare per facilitare l'interazione e la raccolta dei log. Per ulteriori informazioni, consulta [la documentazione del `kubectl ekslogs` plugin].

### Esempio di utilizzo

```
# Collect NodeDiagnostic logs from a single node
kubectl ekslogs <node-name>

# Collect NodeDiagnostic logs from multiple nodes
kubectl ekslogs <node-name-1> <node-name-2> <node-name-3>
```

```
# Collect NodeDiagnostic logs from all nodes with a specific label
kubectl ekslogs -l <key>=<value>
```

## Cattura il traffico di rete su un nodo gestito utilizzando kubectl e S3

Scopri come acquisire il traffico di rete su un nodo gestito Amazon EKS dotato dell'agente di monitoraggio dei nodi. L'agente esegue tcpdump sul nodo, comprime i file di acquisizione e li carica nel tuo bucket S3.

### Prerequisiti

Verifica di disporre di quanto segue:

- Un cluster Amazon EKS Auto Mode esistente con l'agente di monitoraggio dei nodi. Per ulteriori informazioni, consulta the section called "Salute e riparazione dei nodi".
- Lo strumento di riga di comando `kubectl` installato e configurato per comunicare con il cluster.
- La AWS CLI è stata installata e ha effettuato l'accesso con autorizzazioni sufficienti per creare bucket e oggetti S3.
- È installata una versione recente di Python 3.
- L' AWS SDK per Python 3, Boto 3, è installato.
- La libreria PyYAML installata (). `pip install pyyaml`

### Fase 1: Creare di una destinazione bucket S3 (facoltativo)

Se non disponi già di un bucket S3 per archiviare i file di acquisizione, creane uno. Sostituisci *`bucket-name`* e *`region`* con i tuoi valori.

```
aws s3api create-bucket --bucket <bucket-name> \
    --region <region> \
    --create-bucket-configuration LocationConstraint=<region>
```

> **Note**
>
> Il `--create-bucket-configuration` parametro è obbligatorio per tutte le regioni tranne`us-east-1`.

## Fase 2: Avviare l'acquisizione dei pacchetti

Utilizza lo `start-capture.py` script dal repository dell'agente di monitoraggio del nodo (`tools/start-capture.py`) per generare credenziali S3 prefirmate, creare la `NodeDiagnostic` risorsa e applicarla al cluster.

1. Identifica il nodo da cui desideri acquisire il traffico.

```
kubectl get nodes
```

2. Salva lo script start-capture.py dal repository dell'agente di monitoraggio del nodo sul tuo computer locale, quindi eseguilo. Sostituisci *<bucket-name>* e *<node-name>* con i tuoi valori.

```
python3 start-capture.py --bucket <bucket-name> --node <node-name>
```

   Opzioni comuni:

```
# Capture for 5 minutes on eth0 with a filter
python3 start-capture.py --bucket <bucket-name> --node <node-name> \
    --duration 5m --interface eth0 --filter "tcp port 443"

# Preview the YAML without applying
python3 start-capture.py --bucket <bucket-name> --node <node-name> --dry-run
```

   Lo script richiede Python 3 con, `pyyaml` installato `boto3` e `kubectl` configurato per il cluster.

   Lo script genera una `NodeDiagnostic` risorsa come la seguente. Questo esempio viene fornito come riferimento; si noti che i `upload` campi richiedono credenziali S3 POST prefirmate che vengono generate a livello di codice dallo script.

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
  name: <node-name>                    # Required: node instance ID
spec:
  packetCapture:
    duration: "30s"                      # Required: capture duration (max 1h)
    # interface: "eth0"                  # Optional: default is primary ENI. Use
 "any" for all interfaces
    # filter: "tcp port 443"             # Optional: tcpdump filter expression
```

```
    # chunkSizeMB: 10                    # Optional: file rotation size in MB (1-100,
default: 10)
    upload:                               # Required: pre-signed S3 POST credentials
      url: "https://<bucket>.s3.amazonaws.com/"
      fields:
        key: "captures/<node-name>/${filename}"
        # ... other pre-signed POST fields (generated by the script)
```

## Fase 3: Monitora l'avanzamento dell'acquisizione

Controlla lo stato dell'acquisizione.

```
kubectl describe nodediagnostic <node-name>
```

Lo stato mostrerà:

- `Running`mentre l'acquisizione è in corso.
- `Completed`indicando il motivo al `Success` termine dell'acquisizione e al caricamento di tutti i file.
- `Completed`con il motivo `Failure` se l'acquisizione ha riscontrato errori.

Per visualizzare lo stato completo, incluso `captureID` (utilizzato per l'identificazione del percorso S3):

```
kubectl get nodediagnostic <node-name> -o jsonpath='{.status.captureStatuses}'
```

## Passaggio 4: scarica i file di acquisizione da S3

Una volta visualizzato lo stato`Success`, scarica i file di acquisizione da S3.

```
aws s3 cp s3://<bucket-name>/captures/ ./captures/ --recursive
```

I file sono in formato pcap compresso con gzip. Decomprimi e analizza con tcpdump o Wireshark:

```
gunzip captures/*.gz
tcpdump -r captures/capture.pcap0000 -n
```

### Fase 5: rimozione

`NodeDiagnostic`le risorse non vengono eliminate automaticamente. Pulisci dopo aver ottenuto i file di acquisizione. L'eliminazione della risorsa mentre è in corso un'acquisizione interromperà immediatamente l'acquisizione.

```
kubectl delete nodediagnostic <node-name>
```

### Opzioni e comportamento di configurazione

Per il riferimento completo alle `packetCapture` specifiche, alle opzioni di configurazione e ai dettagli sul comportamento, consulta la documentazione sull'acquisizione dei pacchetti nel repository dell'agente di monitoraggio dei nodi.

## Panoramica di Amazon EKS Hybrid Nodes

Con Amazon EKS Hybrid Nodes, puoi utilizzare la tua infrastruttura locale e perimetrale come nodi nei cluster Amazon EKS. AWS gestisce il piano AWS di controllo Kubernetes ospitato del cluster Amazon EKS e tu gestisci i nodi ibridi che vengono eseguiti nei tuoi ambienti locali o edge. Ciò unifica la gestione di Kubernetes in tutti gli ambienti e affida la gestione del piano di controllo di Kubernetes alle applicazioni locali e periferiche. AWS

Amazon EKS Hybrid Nodes funziona con qualsiasi hardware o macchina virtuale on-premises, portando l'efficienza, la scalabilità e la disponibilità di Amazon EKS ovunque sia necessario eseguire le applicazioni. Puoi utilizzare un'ampia gamma di funzionalità di Amazon EKS con Amazon EKS Hybrid Nodes, tra cui componenti aggiuntivi Amazon EKS, Amazon EKS Pod Identity, voci di accesso al cluster, informazioni sui cluster e supporto esteso per le versioni di Kubernetes. Amazon EKS Hybrid Nodes si integra nativamente con AWS servizi tra cui AWS Systems Manager, AWS IAM Roles Anywhere, Amazon Managed Service for Prometheus e Amazon CloudWatch per il monitoraggio centralizzato, la registrazione e la gestione delle identità.

Con Amazon EKS Hybrid Nodes non sono previsti impegni anticipati o tariffe minime e ti viene addebitato un costo orario per le risorse vCPU dei tuoi nodi ibridi quando sono collegati ai cluster Amazon EKS. Per ulteriori informazioni, consulta Prezzi di Amazon EKS.

## Funzionalità

EKS Hybrid Nodes offre le seguenti funzionalità di alto livello:

- Piano di controllo Kubernetes gestito: AWS gestisce il piano di controllo Kubernetes AWS ospitato del cluster EKS e gestisci i nodi ibridi che vengono eseguiti nei tuoi ambienti locali o periferici. Ciò unifica la gestione di Kubernetes in tutti gli ambienti e affida la gestione del piano di controllo di Kubernetes alle applicazioni locali e periferiche. AWS Spostando il piano di controllo Kubernetes su AWS, puoi conservare la capacità locale per le tue applicazioni e avere la certezza che il piano di controllo Kubernetes sia scalabile in base ai tuoi carichi di lavoro.
- Esperienza EKS coerente: la maggior parte delle funzionalità EKS è supportata da EKS Hybrid Nodes per un'esperienza EKS coerente in tutti gli ambienti on-premises e cloud, inclusi componenti aggiuntivi EKS, EKS Pod Identity, voci di accesso al cluster, approfondimenti sui cluster, supporto esteso delle versioni di Kubernetes e altro ancora. Per ulteriori informazioni sui componenti aggiuntivi EKS supportati con EKS Hybrid Nodes, consulta the section called "Configurare i componenti aggiuntivi".
- Osservabilità centralizzata e gestione delle identità: EKS Hybrid Nodes si integra nativamente con servizi tra cui AWS AWS Systems Manager, AWS IAM Roles Anywhere, Amazon Managed

Service for Prometheus e Amazon CloudWatch per il monitoraggio centralizzato, la registrazione e la gestione delle identità.

- Burst-to-cloud o aggiungere capacità locale: un singolo cluster EKS può essere utilizzato per eseguire nodi e nodi ibridi in AWS Regions, AWS Local Zones o AWS Outposts per burst-to-cloud o aggiungere capacità locale ai cluster EKS. Per ulteriori informazioni, consulta Considerations for mixed mode clusters.
- Infrastruttura flessibile: EKS Hybrid Nodes segue un approccio basato sull'infrastruttura personalizzata ed è agnostico rispetto all'infrastruttura utilizzata per i nodi ibridi. Puoi eseguire nodi ibridi su macchine fisiche o virtuali e architetture x86 e ARM, rendendo possibile la migrazione dei carichi di lavoro on-premises in esecuzione su nodi ibridi su diversi tipi di infrastruttura.
- Rete flessibile: con EKS Hybrid Nodes, la comunicazione tra il piano di controllo EKS e i nodi ibridi viene instradata attraverso il VPC e le sottoreti trasferite durante la creazione del cluster, il che si basa sul meccanismo esistente in EKS per il networking dal piano di controllo al nodo. Questo è flessibile rispetto al metodo preferito per connettere le reti locali a un ingresso AWS VPC. Sono disponibili diverse opzioni documentate, tra cui AWS Site-to-Site VPN, AWS Direct Connect o la tua soluzione VPN, e puoi scegliere il metodo più adatto al tuo caso d'uso.

## Limits

- Per ciascun cluster sono supportati fino a 15 CIDR per reti di nodi remoti e 15 CIDR per reti di pod remoti.

## Considerazioni

- EKS Hybrid Nodes può essere utilizzato con cluster EKS nuovi o esistenti.
- EKS Hybrid Nodes è disponibile in tutte le AWS regioni, ad eccezione delle regioni AWS GovCloud (Stati Uniti) e delle regioni della AWS Cina.
- EKS Hybrid Nodes deve disporre di una connessione affidabile tra l'ambiente on-premises e AWS. EKS Hybrid Nodes non è adatto per ambienti disconnessi, interrotti, intermittenti o limitati (DDIL). Se utilizzi un ambiente DDIL, prendi in considerazione Amazon EKS Anywhere. Consulta Best practice for EKS Hybrid Nodes per informazioni sul comportamento dei nodi ibridi nei casi di disconnessione della rete.
- L'esecuzione di EKS Hybrid Nodes sull'infrastruttura cloud, tra cui AWS Regions, AWS Local Zones, AWS Outposts o in altri cloud, non è supportata. Ti verrà addebitata la tariffa per i nodi ibridi se esegui nodi ibridi su istanze Amazon EC2.

- La fatturazione per i nodi ibridi inizia quando i nodi si uniscono al cluster EKS e si interrompe quando i nodi vengono rimossi dal cluster. Assicurati di rimuovere i nodi ibridi dal cluster EKS se non li utilizzi.

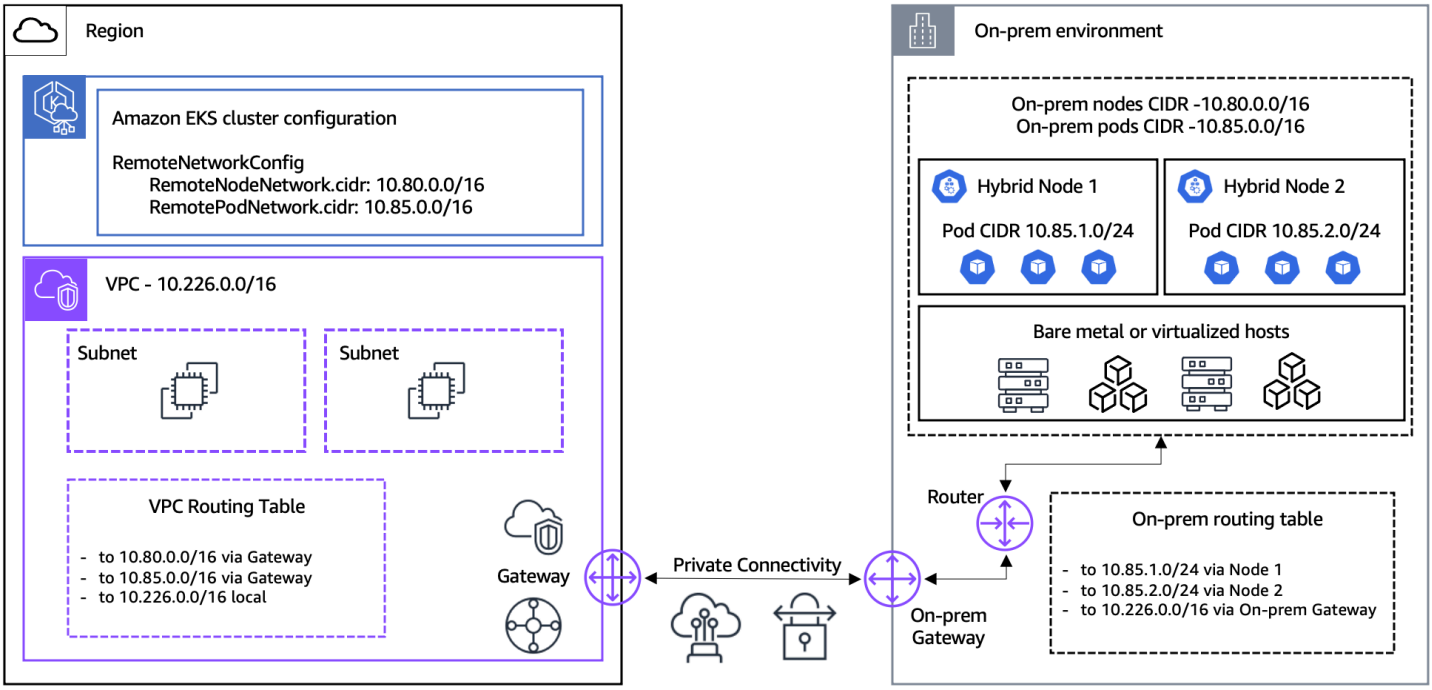
## Risorse aggiuntive

- Workshop EKS Hybrid Nodes: Step-by-step istruzioni per l'implementazione di EKS Hybrid Nodes in un ambiente demo.
- AWS re:Invent: EKS Hybrid Nodes: sessione AWS re:Invent che presenta il lancio dei nodi ibridi EKS con un cliente che mostra come utilizza i nodi ibridi EKS nel proprio ambiente.
- AWS re:Post: Cluster networking for EKS Hybrid Nodes: articolo che spiega vari metodi per configurare la rete per EKS Hybrid Nodes.
- Blog AWS : Run GenAI inference across environments with EKS Hybrid Nodes: post sul blog che mostra come eseguire l'inferenza GenAI in ambienti con EKS Hybrid Nodes.

## Configurazione dei prerequisiti per i nodi ibridi

Per utilizzare Amazon EKS Hybrid Nodes, devi disporre di connettività privata dal tuo ambiente locale da/verso AWS server bare metal o macchine virtuali con un sistema operativo supportato e attivare attivazioni ibride AWS IAM Roles Anywhere o AWS Systems Manager (SSM) configurate. Sei responsabile della gestione di questi prerequisiti durante l'intero ciclo di vita dei nodi ibridi.

- Connettività di rete ibrida dall'ambiente locale da/verso AWS
- Infrastruttura sotto forma di macchine fisiche o virtuali
- Sistema operativo compatibile con i nodi ibridi
- Provider di credenziali IAM on-premises configurato




## Connettività di rete ibrida

La comunicazione tra il piano di controllo di Amazon EKS e i nodi ibridi viene instradata attraverso il VPC e le sottoreti trasferite durante la creazione del cluster, il che si basa sul meccanismo esistente in Amazon EKS per il networking dal piano di controllo al nodo. Sono disponibili diverse opzioni documentate per connettere l'ambiente locale al VPC AWS Site-to-Site , tra cui VPN, AWS Direct Connect o la propria connessione VPN. Consulta le guide utente di AWS Site-to-Site VPN e AWS Direct Connect per ulteriori informazioni su come utilizzare queste soluzioni per la connessione di rete ibrida.

Per un'esperienza ottimale, ti consigliamo di disporre di una connettività di rete affidabile di almeno 100 Mbps e una latenza massima di 200 ms di andata e ritorno per la connessione dei nodi ibridi alla Regione. AWS Questa è una guida generale adatta alla maggior parte dei casi d'uso, ma non è un requisito fondamentale. I requisiti di larghezza di banda e latenza possono variare in base al numero di nodi ibridi e alle caratteristiche del carico di lavoro, come la dimensione dell'immagine dell'applicazione, l'elasticità dell'applicazione, le configurazioni di monitoraggio e registrazione e le dipendenze delle applicazioni dall'accesso ai dati archiviati in altri servizi. AWS Ti consigliamo di eseguire test con applicazioni e ambienti prima di passare all'implementazione per verificare che la configurazione di rete soddisfi i requisiti per i carichi di lavoro.

## Configurazione di reti on-premises

Devi abilitare l'accesso alla rete in entrata dal piano di controllo di Amazon EKS all'ambiente on-premises per consentire al piano di controllo di Amazon EKS di comunicare con il `kubelet` in esecuzione sui nodi ibridi e, facoltativamente, con i webhook in esecuzione sui nodi ibridi. Inoltre, devi abilitare l'accesso alla rete in uscita per i nodi ibridi e i componenti in esecuzione su di essi per comunicare con il piano di controllo di Amazon EKS. Puoi configurare questa comunicazione in modo che rimanga completamente privata con AWS Direct Connect, AWS Site-to-Site VPN o la tua connessione VPN.

Gli intervalli CIDR (Classless Inter-Domain Routing) utilizzati per le reti di nodi e pod locali devono utilizzare IPv4 intervalli di indirizzi RFC-1918 o CGNAT. Il router on-premises deve essere configurato con percorsi verso i nodi locali e, facoltativamente, verso i pod. Consulta the section called "On-premises configurazione di rete" per ulteriori informazioni sui requisiti di rete on-premises, incluso l'elenco completo delle porte e dei protocolli richiesti che devono essere abilitati nel firewall e nell'ambiente on-premises.

## Configurazione del cluster EKS

Per ridurre al minimo la latenza, ti consigliamo di creare il tuo cluster Amazon EKS nella AWS regione più vicina al tuo ambiente locale o edge. Passi il nodo e il pod locali CIDRs durante la creazione del cluster Amazon EKS tramite due campi API: `RemoteNodeNetwork` e`RemotePodNetwork`. Potrebbe essere necessario parlare con il team di rete locale per identificare il nodo e il pod locali. CIDRs Il CIDR dei nodi viene allocato dalla rete on-premises e il CIDR dei pod viene allocato dalla Container Network Interface (CNI) che utilizzi se utilizzi una rete overlay per la tua CNI. Cilium e Calico utilizzano reti overlay per impostazione predefinita.

Il nodo e il pod locali CIDRs che configuri tramite i `RemotePodNetwork` campi `RemoteNodeNetwork` and vengono utilizzati per configurare il piano di controllo di Amazon EKS per indirizzare il traffico attraverso il tuo VPC verso `kubelet` i pod in esecuzione sui tuoi nodi ibridi. Il nodo e il pod locali CIDRs non possono sovrapporsi tra loro, il CIDR VPC che passi durante la creazione del cluster o la configurazione del servizio IPv4 per il tuo cluster Amazon EKS. Inoltre, Pod CIDRs deve essere unico per ogni cluster EKS in modo che il router locale possa instradare il traffico.

Ti consigliamo di utilizzare l'accesso agli endpoint pubblico o privato per l'endpoint del server API Amazon EKS Kubernetes. Se scegli «Pubblico e privato», l'endpoint del server API Amazon EKS Kubernetes verrà sempre convertito in pubblico IPs per i nodi ibridi in esecuzione al di fuori del tuo VPC, il che può impedire ai nodi ibridi di entrare a far parte del cluster. Quando utilizzi l'accesso

pubblico agli endpoint, l'endpoint del server dell'API Kubernetes viene convertito in pubblico IPs e la comunicazione dai nodi ibridi al piano di controllo di Amazon EKS verrà instradata su Internet. Quando scegli l'accesso privato agli endpoint, l'endpoint del server dell'API Kubernetes viene risolto in privato IPs e la comunicazione dai nodi ibridi al piano di controllo di Amazon EKS verrà instradata tramite il tuo collegamento di connettività privato, nella maggior parte dei casi Direct Connect o VPN. AWS AWS Site-to-Site

## Configurazione VPC

Devi configurare il VPC che trasferisci durante la creazione del cluster Amazon EKS con i percorsi nella tabella di routing per il nodo on-premises e, facoltativamente, le reti pod con il gateway privato virtuale (VGW) o il gateway di transito (TGW) come destinazione. Di seguito è riportato un esempio. Sostituisci `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` con i valori per la tua rete on-premises.

| Destinazione | Target | Description |
|---|---|---|
| 10.226.0.0/16 | local | Percorsi del traffico locale verso il VPC all'interno del VPC |
| REMOTE_NODE_CIDR | tgw-abcdef123456 | CIDR nodo on-premises, indirizza il traffico verso il TGW |
| REMOTE_POD_CIDR | tgw-abcdef123456 | CIDR pod on-premises, indirizza il traffico verso il TGW |

## Configurazione del gruppo di sicurezza

Quando crei un cluster, Amazon EKS crea un gruppo di sicurezza denominato `eks-cluster-sg-<cluster-name>-<uniqueID>`. Non puoi modificare le regole in entrata di questo gruppo di sicurezza del cluster, ma puoi limitare le regole in uscita. Devi aggiungere un gruppo di sicurezza aggiuntivo al cluster per abilitare il kubelet e, facoltativamente, i webhook in esecuzione sui nodi ibridi per contattare il piano di controllo di Amazon EKS. Le regole in entrata richieste per questo gruppo di sicurezza aggiuntivo sono riportate di seguito. Sostituisci `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` con i valori per la tua rete on-premises.

| Nome | ID regola del gruppo di sicurezza | Versione IP | Tipo | Protocollo | Intervallo porte | Origine |
|---|---|---|---|---|---|---|
| Nodo on-premises in entrata | sgr-abcdef123456 | IPv4 | HTTPS | TCP | 443 | REMOTE_NODE_CIDR |
| Pod on-premises in entrata | sgr-abcdef654321 | IPv4 | HTTPS | TCP | 443 | REMOTE_POD_CIDR |

## Infrastruttura

Devi disporre di server bare metal o macchine virtuali da utilizzare come nodi ibridi. I nodi ibridi sono indipendenti dall'infrastruttura sottostante e supportano le architetture x86 e ARM. Amazon EKS Hybrid Nodes segue un approccio "a infrastruttura personalizzata", in cui sei responsabile del provisioning e della gestione dei server bare metal o delle macchine virtuali che usi per i nodi ibridi. Sebbene non sia previsto un requisito minimo rigoroso di risorse, ti consigliamo di utilizzare host con almeno 1 vCPU e 1 GB di RAM per i nodi ibridi.

## Sistema operativo

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu e RHEL vengono convalidati su base continuativa per l'uso come sistema operativo di nodi per nodi ibridi. Bottlerocket è supportato solo da AWS ambienti VMware vSphere. AL2023 non è coperto dai piani di AWS supporto se eseguito al di fuori di Amazon EC2. AL2023 può essere utilizzato solo in ambienti virtualizzati locali, consulta la Guida per l'utente di Amazon Linux 2023 per ulteriori informazioni. AWS supporta l'integrazione dei nodi ibridi con i sistemi operativi Ubuntu e RHEL ma non fornisce supporto per il sistema operativo stesso.

Il provisioning e la gestione del sistema operativo sono una tua responsabilità. Quando testi i nodi ibridi per la prima volta, è più semplice eseguire la CLI Amazon EKS Hybrid Nodes (`nodeadm`) su un host già fornito. Per le implementazioni di produzione, ti consigliamo di includere `nodeadm` nelle immagini "gold" del tuo sistema operativo configurato per l'esecuzione come servizio systemd per unire automaticamente gli host ai cluster Amazon EKS all'avvio dell'host.

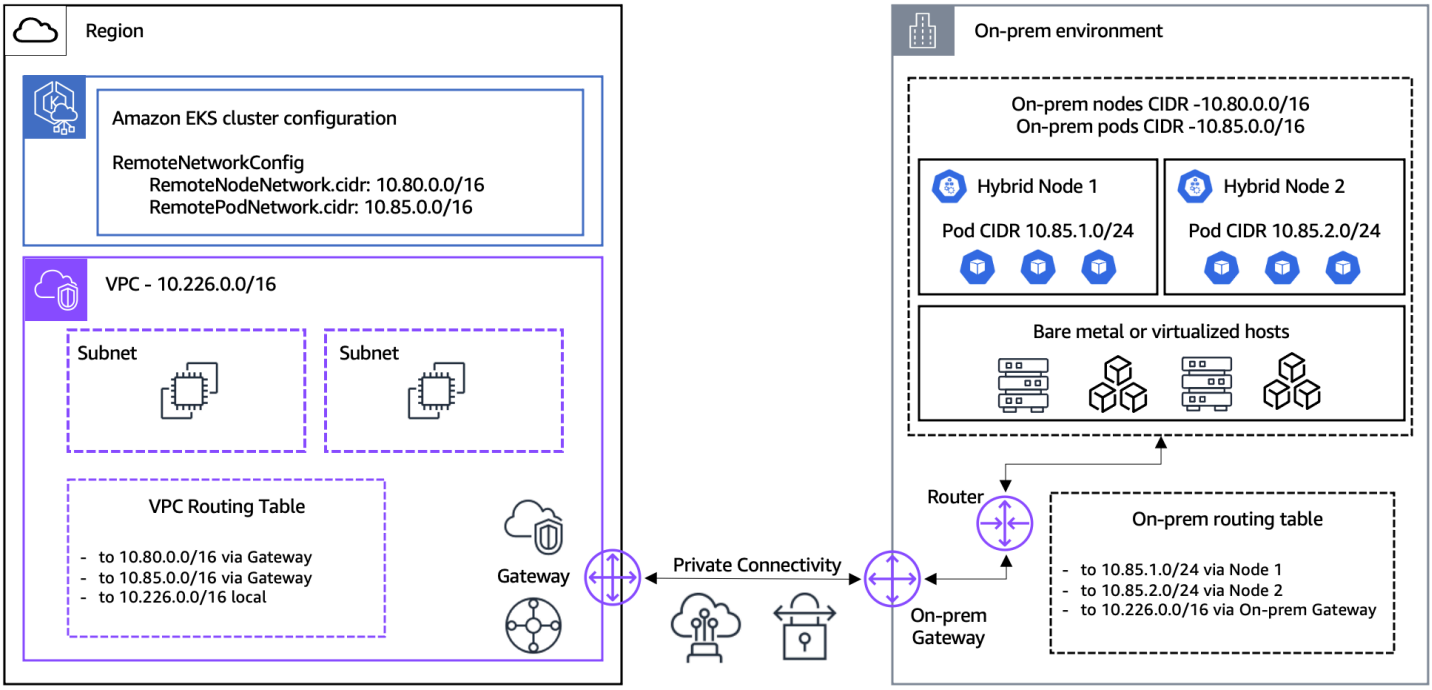
## Provider di credenziali IAM on-premises

I nodi ibridi Amazon EKS utilizzano credenziali IAM temporanee fornite da attivazioni ibride AWS SSM o AWS IAM Roles Anywhere per l'autenticazione con il cluster Amazon EKS. È necessario utilizzare attivazioni ibride AWS SSM o AWS IAM Roles Anywhere con l'Amazon EKS Hybrid Nodes CLI (). `nodeadm` Ti consigliamo di utilizzare le attivazioni ibride AWS SSM se non disponi di un'infrastruttura a chiave pubblica (PKI) esistente con un'autorità di certificazione (CA) e certificati per i tuoi ambienti locali. Se disponi di PKI e certificati esistenti in locale, usa IAM Roles Anywhere. AWS

Analogamente a the section called "Ruolo IAM del nodo" per i nodi in esecuzione su Amazon EC2, creerai un ruolo IAM Hybrid Nodes con le autorizzazioni necessarie per unire nodi ibridi ai cluster Amazon EKS. Se utilizzi AWS IAM Roles Anywhere, configura una policy di fiducia che consenta a AWS IAM Roles Anywhere di assumere il ruolo IAM di Hybrid Nodes e configurare il tuo profilo AWS IAM Roles Anywhere con il ruolo IAM di Hybrid Nodes come ruolo assumibile. Se utilizzi AWS SSM, configura una policy di fiducia che consenta a AWS SSM di assumere il ruolo IAM di Hybrid Nodes e creare l'attivazione ibrida con il ruolo IAM di Hybrid Nodes. Consulta the section called "Preparazione delle credenziali" su come creare il ruolo IAM di Hybrid Nodes con le autorizzazioni richieste.

## Preparazione della rete per i nodi ibridi

Questo argomento fornisce una panoramica della configurazione di rete che devi adottare prima di creare un cluster Amazon EKS e collegare nodi ibridi. Questa guida presuppone che tu abbia soddisfatto i requisiti preliminari per la connettività di rete ibrida utilizzando AWS Site-to-Site VPN, AWS Direct Connect o la tua soluzione VPN.




## On-premises configurazione di rete

### Requisiti minimi di rete

Per un'esperienza ottimale, ti consigliamo di disporre di una connettività di rete affidabile di almeno 100 Mbps e una latenza massima di 200 ms di andata e ritorno per la connessione dei nodi ibridi alla Regione. AWS Questa è una guida generale adatta alla maggior parte dei casi d'uso, ma non è un requisito fondamentale. I requisiti di larghezza di banda e latenza possono variare in base al numero di nodi ibridi e alle caratteristiche del carico di lavoro, come la dimensione dell'immagine dell'applicazione, l'elasticità dell'applicazione, le configurazioni di monitoraggio e registrazione e le dipendenze delle applicazioni dall'accesso ai dati archiviati in altri servizi. AWS Ti consigliamo di eseguire test con applicazioni e ambienti prima di passare all'implementazione per verificare che la configurazione di rete soddisfi i requisiti per i carichi di lavoro.

### On-premises CIDR a nodi e pod

Identifica i CIDR di nodi e pod che utilizzerai per i nodi ibridi e i carichi di lavoro in esecuzione su di essi. Il CIDR dei nodi viene allocato dalla rete on-premises e quello dei pod viene allocato dalla Container Network Interface (CNI) se utilizzi una rete overlay per la CNI. I CIDR dei nodi e i CIDR dei pod on-premises vengono trasmessi come input quando crei il cluster EKS con i campi `RemoteNodeNetwork` e `RemotePodNetwork`. I CIDR dei nodi on-premises devono essere instradabili sulla rete on-premises. Per ulteriori informazioni sull'instradabilità del CIDR dei pod on-premises, consulta la sezione seguente.

Gli intervalli CIDR dei nodi e dei pod on-premises devono soddisfare i seguenti requisiti:

1. Rientrare in uno dei seguenti `IPv4` RFC-1918 intervalli: `10.0.0.0/8` `172.16.0.0/12192.168.0.0/16`, o o all'interno dell'intervallo CGNAT definito da RFC 6598:. `100.64.0.0/10`
2. Non si sovrappongano tra loro, né con il CIDR del VPC per il tuo cluster EKS né con il CIDR `IPv4` del tuo servizio Kubernetes.

> **Important**
>
> Se non specifichi esplicitamente un servizio Kubernetes IPv4 CIDR durante la creazione del cluster, Amazon EKS seleziona automaticamente un servizio CIDR che non sia in conflitto con le reti di nodi e pod remoti configurate. Ciò significa che il servizio assegnato CIDR può differire dai valori predefiniti standard (o). `10.100.0.0/16` `172.20.0.0/16` Ad esempio, se le tue reti remote si sovrappongono`172.20.0.0/16`, Amazon EKS potrebbe invece assegnarle`172.16.0.0/16`. Per evitare assegnazioni inaspettate del servizio CIDR, ti consigliamo di specificare esplicitamente un servizio CIDR IPv4 durante la creazione del cluster. Se hai bisogno di trovare il servizio CIDR assegnato al tuo cluster, esegui. `aws eks describe-cluster --name CLUSTER_NAME --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"`

On-premises routing di rete pod

Quando si utilizza EKS Hybrid Nodes, in genere consigliamo di rendere i CIDR dei pod on-premises instradabili sulla rete on-premises per consentire la comunicazione e la funzionalità complete del cluster tra ambienti cloud e on-premises.

Reti di pod instradabili

Se riesci a rendere la tua rete di pod instradabile su quella on-premises, segui le istruzioni riportate di seguito.

1. Configura il campo `RemotePodNetwork` per il cluster EKS con il CIDR del pod on-premises, le tabelle di routing del VPC con il CIDR del pod on-premises e il gruppo di sicurezza del cluster EKS con il CIDR del pod on-premises.
2. Esistono diverse tecniche che puoi utilizzare per rendere il CIDR del pod on-premises instradabile sulla tua rete on-premises, tra cui Border Gateway Protocol (BGP), route statiche o altre soluzioni

di instradamento personalizzate. BGP è la soluzione consigliata in quanto è più scalabile e facile da gestire rispetto alle soluzioni alternative che richiedono una configurazione del percorso personalizzata o manuale. AWS supporta le funzionalità BGP di Cilium e Calico per la pubblicità dei pod CIDR, vedi e per ulteriori informazioni. the section called "Configurazione della CNI" the section called "CIDR dei pod remoti instradabili"

3. I webhook possono essere eseguiti su nodi ibridi poiché il piano di controllo EKS è in grado di comunicare con gli indirizzi IP dei pod assegnati ai webhook.
4. I carichi di lavoro in esecuzione su nodi cloud sono in grado di comunicare direttamente con i carichi di lavoro in esecuzione su nodi ibridi nello stesso cluster EKS.
5. Altri AWS servizi, come AWS Application Load Balancers e Amazon Managed Service for Prometheus, sono in grado di comunicare con i carichi di lavoro in esecuzione su nodi ibridi per bilanciare il traffico di rete e le metriche dei pod.

Reti di pod non instradabili

Se non riesci a rendere instradabili le reti di pod su quella on-premises, segui le istruzioni riportate di seguito.

1. I webhook non possono essere eseguiti su nodi ibridi perché richiedono la connettività dal piano di controllo EKS agli indirizzi IP dei pod assegnati ai webhook. In questo caso, ti consigliamo di eseguire webhook su nodi cloud nello stesso cluster EKS dei nodi ibridi, consulta the section called "Configurazione di webhook" per ulteriori informazioni.
2. I carichi di lavoro in esecuzione sui nodi cloud non sono in grado di comunicare direttamente con i carichi di lavoro in esecuzione su nodi ibridi quando si utilizza la CNI del VPC per i nodi cloud e Cilium o Calico per i nodi ibridi.
3. Utilizza la Distribuzione del traffico del servizio per mantenere il traffico locale per la zona da cui proviene. Per ulteriori informazioni sulla Distribuzione del traffico del servizio, consulta the section called "Configurazione della distribuzione del traffico del servizio".
4. Configura la CNI per utilizzare la mascheratura in uscita o la Network Address Translation (NAT) per il traffico dei pod in uscita dagli host on-premises. Ciò è abilitato per impostazione predefinita in Cilium. Calico necessita che `natOutgoing` sia impostato su `true`.
5. Altri AWS servizi, come AWS Application Load Balancers e Amazon Managed Service for Prometheus, non sono in grado di comunicare con i carichi di lavoro in esecuzione su nodi ibridi.

Accesso richiesto durante l'installazione e l'aggiornamento del nodo ibrido

Devi avere accesso ai seguenti domini durante il processo di installazione in cui installi le dipendenze dei nodi ibridi sui tuoi host. Questa procedura può essere eseguita una sola volta durante la creazione delle immagini del sistema operativo oppure può essere eseguita su ciascun host in Passaggio di runtime. Ciò include l'installazione iniziale e l'aggiornamento della versione Kubernetes dei nodi ibridi.

Alcuni pacchetti vengono installati utilizzando il gestore di pacchetti predefinito del sistema operativo. Per AL2023 e RHEL viene utilizzato il comando `yum` per installare `containerd`, `ca-certificates`, `iptables` e `amazon-ssm-agent`. Per Ubuntu viene utilizzato `apt` per installare `containerd`, `ca-certificates` e `iptables`, e viene utilizzato `snap` per installare `amazon-ssm-agent`.

| Componente | URL | Protocollo | Porta |
|---|---|---|---|
| Artefatti del nodo EKS (S3) | https://hybrid-assets.eks.amazonaws.com | HTTPS | 443 |
| Endpoint del servizio EKS | https://eks.*region*.amazonaws.com | HTTPS | 443 |
| Endpoint del servizio ECR | https://api.ecr.*region*.amazonaws.com | HTTPS | 443 |
| Endpoint EKS ECR | Consulta the section called "Visualizza i registri delle immagini di Amazon" per gli endpoint regionali. | HTTPS | 443 |
| Endpoint binario SSM [1] | https://amazon-ssm-*region*.s3.*region*.amazonaws.com | HTTPS | 443 |

| Componente | URL | Protocollo | Porta |
|---|---|---|---|
| Endpoint del servizio SSM [1] | https://ssm.*region*.amazonaws.com | HTTPS | 443 |
| Endpoint binario IAM Anywhere [2] | https://rolesanywhere.amazonaws.com | HTTPS | 443 |
| Endpoint del servizio IAM Anywhere [2] | https://rolesanywhere.*region*.amazonaws.com | HTTPS | 443 |
| Endpoint del gestore di pacchetti del sistema operativo | Gli endpoint del Package Repository sono OS-specific e possono variare in base all'area geografica. | HTTPS | 443 |

**Note**

[1] L'accesso agli endpoint AWS SSM è richiesto solo se utilizzi attivazioni ibride AWS SSM per il tuo provider di credenziali IAM locale.
[2] L'accesso agli endpoint AWS IAM è richiesto solo se utilizzi IAM Roles Anywhere per il tuo provider di credenziali AWS IAM locale.

Accesso richiesto per le operazioni in corso del cluster

Per le operazioni in corso del cluster è necessario il seguente accesso alla rete per il firewall on-premises.

**Important**

A seconda della CNI scelta, devi configurare ulteriori regole di accesso alla rete per le porte della CNI. Consulta Cilium documentation e Calico documentation per i dettagli.

| Tipo | Protocollo | Direzione | Porta | Origine | Destinazione | Utilizzo |
|---|---|---|---|---|---|---|
| HTTPS | TCP | In uscita | 443 | CIDR del nodo remoto | IP del cluster EKS [1] | Da kubelet al server API Kubernetes |
| HTTPS | TCP | In uscita | 443 | CIDR del pod remoto | IP del cluster EKS [1] | Da pod al server API Kubernetes |
| HTTPS | TCP | In uscita | 443 | CIDR del nodo remoto | Endpoint del servizio SSM | Aggiornamento delle credenziali delle attivazioni ibride SSM e heartbeat SSM ogni 5 minuti |
| HTTPS | TCP | In uscita | 443 | CIDR del nodo remoto | Endpoint del servizio IAM Anywhere | Aggiornamento delle credenziali di IAM Roles Anywhere |
| HTTPS | TCP | In uscita | 443 | CIDR del pod remoto | Endpoint regionali STS | Da pod a endpoint STS, richiesto solo per IRSA |

| Tipo | Protocollo | Direzione | Porta | Origine | Destinazione | Utilizzo |
|---|---|---|---|---|---|---|
| HTTPS | TCP | In uscita | 443 | CIDR del nodo remoto | Endpoint del servizio di autenticazione Amazon EKS | Da nodo a endpoint di autenticazione Amazon EKS, richiesto solo per Amazon EKS Pod Identity |
| HTTPS | TCP | In entrata | 10250 | IP del cluster EKS [1] | CIDR del nodo remoto | Da server API Kubernetes a kubelet |
| HTTPS | TCP | In entrata | Porte del webhook | IP del cluster EKS [1] | CIDR del pod remoto | Da server API Kubernetes a webhook |

| Tipo | Protocollo | Direzione | Porta | Origine | Destinazione | Utilizzo |
|---|---|---|---|---|---|---|
| HTTPS | TCP, UDP | Inbound, Outbound | 53 | CIDR del pod remoto | CIDR del pod remoto | Da pod a CoreDNS. Se esegui almeno una replica di CoreDNS nel cloud, devi consentire il traffico DNS al VPC su cui è in esecuzione CoreDNS. |
| User-defined | User-defined | Inbound, Outbound | Porte dell'app | CIDR del pod remoto | CIDR del pod remoto | Da pod a pod |

> **Note**
>
> [1] Gli IP del cluster EKS. Consulta la sezione seguente sulle interfacce di rete elastiche di Amazon EKS.

### Interfacce di rete elastiche di Amazon EKS

Amazon EKS collega le interfacce di rete alle sottoreti del VPC che passi durante la creazione del cluster per abilitare la comunicazione tra il piano di controllo EKS e il tuo VPC. Le interfacce di rete create da Amazon EKS possono essere trovate dopo la creazione del cluster nella console Amazon EC2 o con l'interfaccia a riga di comando. AWS Le interfacce di rete originali vengono eliminate e vengono create di nuove quando vengono applicate modifiche al cluster EKS, ad esempio gli aggiornamenti delle versioni di Kubernetes. Puoi limitare l'intervallo IP per le interfacce di rete

Amazon EKS utilizzando dimensioni di sottorete vincolate per le sottoreti che passate durante la creazione del cluster, il che semplifica la configurazione del firewall locale per consentire la inbound/outbound connettività a questo set di IP noto e vincolato. Per verificare in quali sottoreti vengono create le interfacce di rete, puoi limitare il numero di sottoreti specificate a due quando crei un cluster o puoi aggiornare le sottoreti dopo aver creato il cluster.

Le interfacce di rete fornite da Amazon EKS hanno una descrizione del formato `Amazon EKS your-cluster-name` . Guarda l'esempio seguente per un comando AWS CLI che puoi usare per trovare gli indirizzi IP delle interfacce di rete fornite da Amazon EKS. Sostituisci `VPC_ID` con l'ID del VPC che passi durante la creazione del cluster.

```
aws ec2 describe-network-interfaces \
--query 'NetworkInterfaces[?(VpcId == VPC_ID && contains(Description,Amazon
 EKS))].PrivateIpAddress'
```

AWS Configurazione di VPC e sottorete

Ai cluster con nodi ibridi si applicano gli attuali Requisiti per VPC e sottoreti per Amazon EKS. Inoltre, il VPC del tuo CIDR non può sovrapporsi ai CIDR dei nodi e dei pod on-premises. Devi configurare i percorsi nella tabella di routing VPC per i CIDR dei nodi on-premises e, facoltativamente, dei pod. Questi percorsi devono essere configurati per indirizzare il traffico verso il gateway utilizzato per la connettività di rete ibrida, che in genere è un gateway privato virtuale (VGW) o un gateway di transito (TGW). Se utilizzi TGW o VGW per connettere il tuo VPC all'ambiente on-premises, devi creare un allegato TGW o VGW per il tuo VPC. Il VPC deve supportare il nome host DNS e la risoluzione DNS.

I passaggi seguenti utilizzano la AWS CLI. Puoi anche creare queste risorse in Console di gestione AWS o con altre interfacce come AWS CDK o AWS CloudFormation Terraform.

Passaggio 1: creazione di un VPC

1. Per creare un nuovo VPC, esegui il comando seguente. Sostituisci VPC_CIDR con un intervallo CIDR IPv4 che sia RFC 1918 (privato), CGNAT (RFC 6598) o non RFC -CGNAT (pubblico) (ad esempio, 10.0.0. 1918/non 0/16). Nota: la risoluzione DNS, che è un requisito EKS, è abilitata per impostazione predefinita per il VPC.

   ```
   aws ec2 create-vpc --cidr-block VPC_CIDR
   ```

2. Abilita i nomi degli host DNS per il tuo VPC. Ricorda, la risoluzione DNS è abilitata per impostazione predefinita per il VPC. Sostituisci `VPC_ID` con l'ID del VPC creato nel passaggio precedente.

```
aws ec2 modify-vpc-attribute --vpc-id VPC_ID --enable-dns-hostnames
```

Passaggio 2: creazione di sottoreti

Crea almeno due sottoreti. Amazon EKS utilizza queste sottoreti per le interfacce di rete del cluster. Per maggiori informazioni, consulta Subnets requirements and considerations.

1. È possibile trovare le zone di disponibilità per una AWS regione con il seguente comando. Sostituisci `us-west-2` con la tua regione.

```
aws ec2 describe-availability-zones \
     --query 'AvailabilityZones[?(RegionName == us-west-2)].ZoneName'
```

2. Crea una sottorete. Sostituisci `VPC_ID` con l'ID del VPC. Sostituisci `SUBNET_CIDR` con il blocco CIDR per la tua sottorete (ad esempio 10.0.1). 0/24 ). Sostituisci `AZ` con la zona di disponibilità in cui verrà creata la sottorete (ad esempio us-west-2a). Le sottoreti che crei devono trovarsi in almeno due zone di disponibilità differenti.

```
aws ec2 create-subnet \
    --vpc-id VPC_ID \
    --cidr-block SUBNET_CIDR \
    --availability-zone AZ
```

(Facoltativo) Fase 3: collegare il VPC con Amazon VPC Transit Gateway (TGW) oppure AWS Gateway privato virtuale (VGW) Direct Connect

Se utilizzi un TGW o un VGW, collega il tuo VPC al TGW o al VGW. Per ulteriori informazioni, consulta Amazon VPC attachments in Amazon VPC Transit Gateways o AWS Direct Connect virtual private gateway associations.

Gateway di transito

Esegui il comando seguente per collegare un gateway di transito. Sostituisci `VPC_ID` con l'ID del VPC. Sostituisci `SUBNET_ID1` e `SUBNET_ID2` con gli ID delle sottoreti create nel passaggio precedente. Sostituisci `TGW_ID` con l'ID del TGW.

```
aws ec2 create-transit-gateway-vpc-attachment \
    --vpc-id VPC_ID \
    --subnet-ids SUBNET_ID1 SUBNET_ID2 \
    --transit-gateway-id TGW_ID
```

Gateway virtuale privato

Esegui il comando seguente per collegare un Gateway di transito. Sostituisci `VPN_ID` con l'ID del VGW. Sostituisci `VPC_ID` con l'ID del VPC.

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id VPN_ID \
    --vpc-id VPC_ID
```

(Facoltativo) Passaggio 4: creazione di una tabella di routing

Puoi modificare la tabella di routing principale per il VPC o creare una tabella di routing personalizzata. I passaggi seguenti creano una tabella di routing personalizzata con le rotte verso i CIDR di nodi e pod on-premises. Per ulteriori informazioni, consulta le Tabelle di routing delle sottoreti. Sostituisci `VPC_ID` con l'ID del VPC.

```
aws ec2 create-route-table --vpc-id VPC_ID
```

Passaggio 5: creazione di percorsi per nodi e pod on-premises

Crea percorsi nella tabella di routing per ciascuno dei tuoi nodi remoti on-premises. Puoi modificare la tabella di routing principale per il VPC o utilizzare la tabella di routing personalizzata creata nel passaggio precedente.

Gli esempi seguenti mostrano come creare percorsi per i CIDR di nodi e pod on-premises. Negli esempi viene utilizzato un gateway di transito (TGW) per connettere il VPC all'ambiente on-premises. Se disponi di più CIDR di nodi e pod on-premises, ripeti i passaggi per ciascun CIDR.

- Se utilizzi un gateway Internet o un gateway privato virtuale (VGW), sostituisci `--transit-gateway-id` con `--gateway-id`.
- Sostituisci `RT_ID` con l'ID della tabella di routing creata nel passaggio precedente.
- Sostituisci `REMOTE_NODE_CIDR` con l'intervallo CIDR che utilizzerai per i tuoi nodi ibridi.
- Sostituisci `REMOTE_POD_CIDR` con l'intervallo CIDR che utilizzerai per i pod in esecuzione su nodi ibridi. L'intervallo CIDR dei pod corrisponde alla configurazione Container Networking Interface (CNI), che comunemente utilizza una rete overlay on-premises. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".
- Sostituisci `TGW_ID` con l'ID del TGW.

Rete di nodi remoti

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_NODE_CIDR \
    --transit-gateway-id TGW_ID
```

Rete di pod remoti

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_POD_CIDR \
    --transit-gateway-id TGW_ID
```

(Facoltativo) Passaggio 6: associazione delle sottoreti alla tabella di routing

Se hai creato una tabella di routing personalizzata nel passaggio precedente, associa ciascuna delle sottoreti che hai creato nel passaggio precedente alla tua tabella di routing personalizzata. Se stai modificando la tabella di routing principale del VPC, le sottoreti vengono automaticamente associate alla tabella di routing principale del VPC e puoi saltare questo passaggio.

Esegui questo comando per ciascuna sottorete creata nei passaggi precedenti. Sostituisci `RT_ID` con la tabella di routing creata nel passaggio precedente. Sostituisci `SUBNET_ID` con l'ID di una sottorete.

```
aws ec2 associate-route-table --route-table-id RT_ID --subnet-id SUBNET_ID
```

### Configurazione del gruppo di sicurezza del cluster

Il seguente accesso per il gruppo di sicurezza del cluster EKS è necessario per le operazioni in corso del cluster. Amazon EKS crea automaticamente le regole dei gruppi di sicurezza in entrata richieste per i nodi ibridi quando crei o aggiorni il cluster con reti di nodi e pod remoti configurate. Poiché i gruppi di sicurezza consentono tutto il traffico in uscita per impostazione predefinita, Amazon EKS non modifica automaticamente le regole in uscita del gruppo di sicurezza del cluster per i nodi ibridi. Se desideri personalizzare il gruppo di sicurezza del cluster, puoi limitare il traffico alle regole riportate nella tabella seguente.

| Tipo | Protocollo | Direzione | Porta | Origine | Destinazione | Utilizzo |
|---|---|---|---|---|---|---|
| HTTPS | TCP | In entrata | 443 | CIDR del nodo remoto | N/A | Da Kubelet al server API Kubernetes |
| HTTPS | TCP | In entrata | 443 | CIDR del pod remoto | N/A | Pod che richiedono l'accesso al server API K8s quando il CNI non utilizza NAT per il traffico dei pod. |
| HTTPS | TCP | In uscita | 10250 | N/A | CIDR del nodo remoto | Da server API Kubernetes a Kubelet |
| HTTPS | TCP | In uscita | Porte del webhook | N/A | CIDR del pod remoto | Da server API Kubernetes |

| Tipo | Protocollo | Direzione | Porta | Origine | Destinazione | Utilizzo |
|---|---|---|---|---|---|---|
| | | | | | | a webhook (se esegui webhook su nodi ibridi) |

> **Important**
>
> Limiti delle regole dei gruppi di sicurezza: per impostazione predefinita, i gruppi di sicurezza di Amazon EC2 hanno un massimo di 60 regole in entrata. Le regole in entrata del gruppo di sicurezza potrebbero non essere applicabili se il gruppo di sicurezza del cluster si avvicina a questo limite. In questo caso, potrebbe essere necessario aggiungere manualmente le regole in entrata mancanti.
> Responsabilità di pulizia CIDR: se rimuovi reti di nodi o pod remoti dai cluster EKS, quest'ultimo non rimuove automaticamente le regole del gruppo di sicurezza corrispondenti. È responsabilità dell'utente rimuovere manualmente le reti di nodi o pod remoti non utilizzate dalle regole del gruppo di sicurezza.

Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza".

(Facoltativo) Configurazione manuale del gruppo di sicurezza

Se devi creare gruppi di sicurezza aggiuntivi o modificare le regole create automaticamente, puoi utilizzare i seguenti comandi come riferimento. Per impostazione predefinita, il comando seguente crea un gruppo di sicurezza che consente tutti gli accessi in uscita. Puoi limitare l'accesso in uscita in modo da includere solo le regole precedenti. Se stai pianificando una limitazione delle regole, ti consigliamo di testare accuratamente la connettività delle applicazioni e dei pod prima di applicare le regole modificate a un cluster di produzione.

- Nel primo comando, sostituisci `SG_NAME` con un nome per il tuo gruppo di sicurezza
- Nel primo comando, sostituisci `VPC_ID` con l'ID del VPC creato nel passaggio precedente

- Nel secondo comando, sostituisci `SG_ID` con l'ID del gruppo di sicurezza creato nel primo comando
- Nel secondo comando, sostituisci `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` con i valori per i nodi ibridi e la rete on-premises.

```
aws ec2 create-security-group \
    --group-name SG_NAME \
    --description "security group for hybrid nodes" \
    --vpc-id VPC_ID
```

```
aws ec2 authorize-security-group-ingress \
    --group-id SG_ID \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort": 443,
 "IpRanges": [{"CidrIp": "REMOTE_NODE_CIDR"}, {"CidrIp": "REMOTE_POD_CIDR"}]}]'
```

## Preparazione del sistema operativo per i nodi ibridi

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu e RHEL vengono convalidati su base continuativa per l'uso come sistema operativo di nodi per nodi ibridi. Bottlerocket è supportato solo AWS negli ambienti VMware vSphere. AL2023 non è coperto dai piani di AWS supporto se eseguito al di fuori di Amazon EC2. AL2023 può essere utilizzato solo in ambienti virtualizzati locali, consulta la Amazon Linux 2023 User Guide per ulteriori informazioni. AWS supporta l'integrazione dei nodi ibridi con i sistemi operativi Ubuntu e RHEL ma non fornisce supporto per il sistema operativo stesso.

Il provisioning e la gestione del sistema operativo sono una tua responsabilità. Quando testi i nodi ibridi per la prima volta, è più semplice eseguire la CLI Amazon EKS Hybrid Nodes (`nodeadm`) su un host già fornito. Per le implementazioni di produzione, ti consigliamo di includere `nodeadm` nelle immagini del tuo sistema operativo configurate per l'esecuzione come servizio systemd per unire automaticamente gli host ai cluster Amazon EKS all'avvio dell'host. Se utilizzi Bottlerocket come sistema operativo del nodo su vSphere, non è necessario utilizzare `nodeadm` poiché Bottlerocket contiene già le dipendenze necessarie per i nodi ibridi e si connetterà automaticamente al cluster configurato all'avvio dell'host.

⚠ Versione node-adm richiesta per il provider di credenziali SSM

Se si utilizza AWS Systems Manager (SSM) come provider di credenziali per i nodi ibridi, è necessario utilizzare la `nodeadm` versione `1.0.19` o successiva per le nuove installazioni e

gli aggiornamenti. Le versioni precedenti di `nodeadm` contengono una chiave di firma SSM obsoleta e hanno esito negativo durante `nodeadm install` e `nodeadm upgrade` con il seguente errore di verifica della firma:

```
"msg":"Command failed","error":"failed to install ssm installer: validating ssm-setup-cli signature: Signature Verification Error: No matching signature"
```

Per risolvere questo errore, scarica la versione più recente di `nodeadm` prima di eseguire `nodeadm install` o. `nodeadm upgrade`

### Compatibilità delle versioni

La tabella seguente rappresenta le versioni del sistema operativo compatibili e convalidate per l'uso come sistema operativo dei nodi per i nodi ibridi. Se si utilizzano altre varianti o versioni del sistema operativo non incluse in questa tabella, la compatibilità dei nodi ibridi con la variante o la versione del sistema operativo in uso non è coperta dall' AWS assistenza. I nodi ibridi sono indipendenti dall'infrastruttura sottostante e supportano le architetture x86 e ARM.

| Sistema operativo | Versioni |
|---|---|
| Amazon Linux | Amazon Linux 2023 (AL2023) |
| Bottlerocket | La versione 1.37.0 e le versioni successive VMware che eseguono Kubernetes 1.28 e versioni successive |
| Ubuntu | Ubuntu 20.04, Ubuntu 22.04, Ubuntu 24.04 |
| Red Hat Enterprise Linux | RHEL 8, RHEL 9 |

### Considerazioni sul sistema operativo

#### Ambito generale

- La CLI Amazon EKS Hybrid Nodes (`nodeadm`) può essere utilizzata per semplificare l'installazione e la configurazione dei componenti e delle dipendenze dei nodi ibridi. Puoi eseguire il processo `nodeadm install` durante le pipeline di creazione dell'immagine del sistema operativo o in

Passaggio di runtime su ogni host on-premises. Per ulteriori informazioni sui componenti che `nodeadm` installa, consulta the section called "nodeadm dei nodi ibridi".

- Se utilizzi un proxy nell'ambiente on-premises per accedere a Internet, è necessaria una configurazione aggiuntiva del sistema operativo per i processi di installazione e aggiornamento per configurare il gestore di pacchetti per l'utilizzo del proxy. Per istruzioni, consulta the section called "Configurazione del proxy".

Razzo a bottiglia

- I passaggi e gli strumenti per connettere un nodo Bottlerocket sono diversi da quelli per altri sistemi operativi e sono descritti separatamente in the section called "Connessione dei nodi ibridi con Bottlerocket", anziché nelle istruzioni di the section called "Connessione di nodi ibridi".
- I passaggi per Bottlerocket non utilizzano lo strumento CLI dei nodi ibridi, `nodeadm`.
- Solo le varianti VMware della versione Bottlerocket 1.37.0 e successive sono supportate con EKS Hybrid Nodes. Le varianti VMware di Bottlerocket sono disponibili per le versioni 1.28 e successive di Kubernetes. Altre varianti di Bottlerocket non sono supportate come sistema operativo dei nodi ibridi. NOTA: le varianti VMware di Bottlerocket sono disponibili solo per l'architettura x86_64.

Containerd

- Containerd è il runtime del container Kubernetes standard ed è una dipendenza per i nodi ibridi, nonché per tutti i tipi di calcolo dei nodi Amazon EKS. La CLI di Amazon EKS Hybrid Nodes (`nodeadm`) tenta di installare containerd durante il processo `nodeadm install`. È possibile configurare l'installazione containerd nella Passaggio di runtime di `nodeadm install` con l'opzione `--containerd-source` della riga di comando. Le opzioni valide sono `none`, `distro` e `docker`. Se stai usando RHEL, `distro` non è un'opzione valida e puoi configurare `nodeadm` per installare la build containerd dai repository di Docker oppure puoi installare containerd manualmente. Quando utilizzi AL2023 o Ubuntu, `nodeadm` installa containerd per impostazione predefinita dalla distribuzione del sistema operativo. Se non vuoi che nodeadm installi containerd, usa l'opzione `--containerd-source none`.

Ubuntu

- Se stai usando Ubuntu 24.04, potresti dover aggiornare la tua versione di containerd o modificare la AppArmor configurazione per adottare una correzione che consenta ai pod di terminare correttamente, vedi Ubuntu #2065423. https://bugs.launchpad.net/ubuntu/+source/containerd-app/

\+bug/2065423 È necessario un riavvio per applicare le modifiche al profilo. AppArmor L'ultima versione di Ubuntu 24.04 ha una versione containerd aggiornata nel suo gestore di pacchetti con la correzione (versione containerd 1.7.19 o successive).

ARM

- Se si utilizza hardware ARM, è necessario un processore ARMv8.2 compatibile con la Cryptography Extension (ARMv8.2+crypto) per eseguire la versione 1.31 e successive del componente aggiuntivo EKS kube-proxy. Tutti i sistemi Raspberry Pi precedenti al Raspberry Pi 5, così come i processori basati, non soddisfano questo requisito. Cortex-A72 Come soluzione alternativa, puoi continuare a utilizzare la versione 1.30 del componente aggiuntivo EKS kube-proxy fino alla fine del supporto esteso a luglio del 2026, consultare il Calendario delle versioni di Kubernetes o utilizzare un'immagine kube-proxy personalizzata dall'upstream.
- Il seguente messaggio di errore nel registro kube-proxy indica questa incompatibilità:

```
Fatal glibc error: This version of Amazon Linux requires a newer ARM64 processor
 compliant with at least ARM architecture 8.2-a with Cryptographic extensions. On EC2
 this is Graviton 2 or later.
```

Creazione di immagini del sistema operativo

Amazon EKS fornisce modelli di Packer di esempio che puoi utilizzare per creare immagini del sistema operativo che includono `nodeadm` e lo configurano per l'esecuzione all'avvio dell'host. Questa procedura è consigliata per evitare di trasferire le dipendenze dei nodi ibridi singolarmente su ciascun host e per automatizzare il processo di bootstrap dei nodi ibridi. Puoi utilizzare i modelli Packer di esempio con un'immagine ISO di Ubuntu 22.04, Ubuntu 24.04, RHEL 8 o RHEL 9 e produrre immagini con questi formati: OVA, Qcow2 o raw.

Prerequisiti

Prima di utilizzare i modelli Packer di esempio, è necessario che sul computer da cui esegui Packer sia installato quanto segue.

- Packer versione 1.11.0 o successiva. Per istruzioni sull'installazione di Packer, consulta Install Packer nella documentazione di Packer.
- Se stai creando OVA, VMware vSphere plug-in 1.4.0 o successivo
- Se stai creando immagini `Qcow2` o grezze, plug-in QEMU versione 1.x

Impostazione delle variabili di ambiente

Prima di eseguire la build di Packer, imposta le seguenti variabili di ambiente sul computer da cui stai eseguendo Packer.

Ambito generale

Le seguenti variabili di ambiente devono essere impostate per creare immagini con tutti i sistemi operativi e i formati di output.

| Variabile di ambiente | Tipo | Description |
|---|---|---|
| PKR_SSH_PASSWORD | Stringa | Packer utilizza le variabili `ssh_username` e `ssh_password` di SSH nella macchina creata durante il provisioning. Ciò deve corrispondere alle password utilizzate per creare l'utente iniziale all'interno dei file kickstart o di dati utente del rispettivo sistema operativo. L'impostazione predefinita è "builder" o "ubuntu" a seconda del sistema operativo. Quando imposti la password, assicurati di cambiarla all'interno del file `ks.cfg` o `user-data` corrispondente. |
| ISO_URL | Stringa | URL dell'ISO da utilizzare. Può essere un collegamento web da scaricare da un server o un percorso assoluto verso un file locale |
| ISO_CHECKSUM | Stringa | Checksum associato per l'ISO fornito. |

| Variabile di ambiente | Tipo | Description |
| --- | --- | --- |
| CREDENTIAL_PROVIDER | Stringa | Provider di credenziali per nodi ibridi. I valori validi sono `ssm` (impostazione predefinita) per le attivazioni ibride SSM e `iam` per IAM Roles Anywhere |
| K8S_VERSION | Stringa | Versione di Kubernetes per nodi ibridi (ad esempio `1.31`). Per le versioni di Kubernetes supportate, consulta la pagina Amazon EKS supported versions. |
| NODEADM_ARCH | Stringa | Architettura per `nodeadm install`. Seleziona `amd` oppure `arm`. |

RHEL

Se utilizzi RHEL, è necessario impostare le seguenti variabili di ambiente.

| Variabile di ambiente | Tipo | Description |
| --- | --- | --- |
| RH_USERNAME | Stringa | nome utente del gestore di sottoscrizioni RHEL |
| RH_PASSWORD | Stringa | password del gestore delle sottoscrizioni RHEL |
| RHEL_VERSION | Stringa | Versione iso di Rhel in uso. I valori validi sono 8 e 9. |

Ubuntu

Non sono richieste variabili di Ubuntu-specific ambiente.

vSphere

Se stai creando un OVA VMware vSphere, è necessario impostare le seguenti variabili di ambiente.

| Variabile di ambiente | Tipo | Description |
|---|---|---|
| VSPHERE_SERVER | Stringa | Indirizzo del server vSphere |
| VSPHERE_USER | Stringa | Nome utente vSphere |
| VSPHERE_PASSWORD | Stringa | Password vSphere |
| VSPHERE_DATACENTER | Stringa | Nome del data center vSphere |
| VSPHERE_CLUSTER | Stringa | Nome del cluster vSphere |
| VSPHERE_DATASTORE | Stringa | Nome del datastore vSphere |
| VSPHERE_NETWORK | Stringa | Nome della rete vSphere |
| VSPHERE_OUTPUT_FOLDER | Stringa | Cartella di output vSphere per i modelli |

QEMU

| Variabile di ambiente | Tipo | Description |
|---|---|---|
| PACKER_OUTPUT_FORMAT | Stringa | Formato di output per il generatore QEMU. I valori validi sono `qcow2` e `raw`. |

Convalida del modello

Prima di eseguire la build, convalida il modello con il seguente comando dopo aver impostato le variabili di ambiente. Sostituisci `template.pkr.hcl` se stai usando un nome diverso per il tuo modello.

```
packer validate template.pkr.hcl
```

Creazione di immagini

Crea le tue immagini con i seguenti comandi e usa il flag `-only` per specificare la destinazione e il sistema operativo delle tue immagini. Sostituisci `template.pkr.hcl` se stai usando un nome diverso per il tuo modello.

OVA vSphere

> **Note**
>
> Se utilizzi RHEL con vSphere, devi convertire i file kickstart in un'immagine OEMDRV e passarla come ISO da cui avviare il sistema. Per ulteriori informazioni, consultate il Readme di Packer nell'EKS Hybrid Nodes GitHub Repository.

OVA Ubuntu 22.04

```
packer build -only=general-build.vsphere-iso.ubuntu22 template.pkr.hcl
```

OVA Ubuntu 24.04

```
packer build -only=general-build.vsphere-iso.ubuntu24 template.pkr.hcl
```

OVA RHEL 8

```
packer build -only=general-build.vsphere-iso.rhel8 template.pkr.hcl
```

OVA RHEL 9

```
packer build -only=general-build.vsphere-iso.rhel9 template.pkr.hcl
```

QEMU

> **Note**
>
> Se stai creando un'immagine per una CPU host specifica che non corrisponde all'host del tuo generatore, consulta la documentazione QEMU per cercare il nome che corrisponde alla tua CPU host e usa il flag `-cpu` con il nome della CPU host quando esegui i seguenti comandi.

Qcow2/Raw Ubuntu 22.04

```
packer build -only=general-build.qemu.ubuntu22 template.pkr.hcl
```

Qcow2/Raw Ubuntu 24.04

```
packer build -only=general-build.qemu.ubuntu24 template.pkr.hcl
```

Qcow2/Raw RHEL 8

```
packer build -only=general-build.qemu.rhel8 template.pkr.hcl
```

Qcow2/Raw RHEL 9

```
packer build -only=general-build.qemu.rhel9 template.pkr.hcl
```

Trasferimento della configurazione di nodeadm tramite i dati utente

Puoi trasferire la configurazione per `nodeadm` ai tuoi dati utente tramite cloud-init per configurare e connettere automaticamente i nodi ibridi al cluster EKS all'avvio dell'host. Di seguito è riportato un esempio di come eseguire questa operazione quando utilizzi VMware vSphere come infrastruttura per i nodi ibridi.

1. Installa la `govc` CLI seguendo le istruzioni nel file readme di govc on. https://github.com/vmware/govmomi/blob/main/govc/README.md GitHub
2. Dopo aver eseguito la build di Packer nella sezione precedente e aver effettuato il provisioning del modello, puoi clonare il modello per creare più nodi diversi utilizzando quanto segue. Devi clonare il modello per ogni nuova macchina virtuale che stai creando e che verrà utilizzata per i nodi ibridi. Sostituisci le variabili nel comando seguente con i valori per il tuo ambiente. Il comando `VM_NAME` riportato di seguito viene utilizzato come `NODE_NAME` quando inserisci i nomi delle macchine virtuali tramite il file `metadata.yaml`.

   ```
   govc vm.clone -vm "/PATH/TO/TEMPLATE" -ds="YOUR_DATASTORE" \
       -on=false -template=false -folder=/FOLDER/TO/SAVE/VM "VM_NAME"
   ```
3. Dopo aver clonato il modello per ciascuna delle tue nuove macchine virtuali, crea un `userdata.yaml` e un `metadata.yaml` per le tue macchine virtuali. Le macchine virtuali possono condividerle gli stessi `userdata.yaml` e `metadata.yaml` e le popolerai singolarmente nei passaggi seguenti. La configurazione di `nodeadm` viene creata e definita nella sezione

`write_files` del tuo `userdata.yaml`. L'esempio seguente utilizza le attivazioni ibride AWS SSM come provider di credenziali locale per i nodi ibridi. Per ulteriori informazioni sulla configurazione di `nodeadm`, consulta the section called "nodeadm dei nodi ibridi".

userdata.yaml:

```
#cloud-config
users:
  - name: # username for login. Use 'builder' for RHEL or 'ubuntu' for Ubuntu.
    passwd: # password to login. Default is 'builder' for RHEL.
    groups: [adm, cdrom, dip, plugdev, lxd, sudo]
    lock-passwd: false
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash

write_files:
  - path: /usr/local/bin/nodeConfig.yaml
    permissions: '0644'
    content: |
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
          cluster:
              name: # Cluster Name
              region: # AWS region
          hybrid:
              ssm:
                  activationCode: # Your ssm activation code
                  activationId: # Your ssm activation id

runcmd:
  - /usr/local/bin/nodeadm init -c file:///usr/local/bin/nodeConfig.yaml >> /var/log/
nodeadm-init.log 2>&1
```

metadata.yaml:

Crea un `metadata.yaml` per il tuo ambiente. Mantieni il formato della variabile `"$NODE_NAME"` nel file poiché questo verrà compilato con valori in un passaggio successivo.

```
instance-id: "$NODE_NAME"
local-hostname: "$NODE_NAME"
network:
```

```
  version: 2
  ethernets:
    nics:
      match:
        name: ens*
      dhcp4: yes
```

4. Aggiungi i file `userdata.yaml` e `metadata.yaml` come stringhe `gzip+base64` con i seguenti comandi. I seguenti comandi devono essere eseguiti per ciascuna delle macchine virtuali che si stanno creando. Sostituisci `VM_NAME` con il nome della macchina virtuale che stai aggiornando.

```
export NODE_NAME="VM_NAME"
export USER_DATA=$(gzip -c9 <userdata.yaml | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata="${USER_DATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata.encoding=gzip+base64

envsubst '$NODE_NAME' < metadata.yaml > metadata.yaml.tmp
export METADATA=$(gzip -c9 <metadata.yaml.tmp | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata="${METADATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata.encoding=gzip+base64
```

5. Accendi le tue nuove macchine virtuali, che dovrebbero connettersi automaticamente al cluster EKS che hai configurato.

```
govc vm.power -on "${NODE_NAME}"
```

## Preparazione delle credenziali per i nodi ibridi

I nodi ibridi Amazon EKS utilizzano credenziali IAM temporanee fornite da attivazioni ibride AWS SSM o AWS IAM Roles Anywhere per autenticarsi con il cluster Amazon EKS. È necessario utilizzare le attivazioni ibride AWS SSM o AWS IAM Roles Anywhere con la CLI () di Amazon EKS Hybrid Nodes. `nodeadm` Non dovresti usare sia le attivazioni ibride AWS SSM che IAM Roles Anywhere. AWS Ti consigliamo di utilizzare le attivazioni ibride AWS SSM se non disponi di un'infrastruttura a

chiave pubblica (PKI) esistente con un'autorità di certificazione (CA) e certificati per i tuoi ambienti locali. Se disponi di PKI e certificati esistenti in locale, utilizza IAM Roles Anywhere. AWS

Ruolo IAM dei nodi ibridi

Prima di poter connettere i nodi ibridi al cluster Amazon EKS, devi creare un ruolo IAM che verrà utilizzato con le attivazioni ibride AWS SSM o AWS IAM Roles Anywhere per le credenziali dei nodi ibridi. Dopo la creazione del cluster, utilizzerai questo ruolo con una voce di accesso Amazon EKS o una `aws-auth` ConfigMap voce per mappare il ruolo IAM su Kubernetes Role-Based Access Control (RBAC). Per ulteriori informazioni sull'associazione del ruolo IAM dei nodi ibridi con il RBAC di Kubernetes, consulta the section called "Preparazione dell'accesso ai cluster".

Il ruolo IAM dei nodi ibridi deve disporre delle seguenti autorizzazioni.

- Autorizzazioni per `nodeadm` utilizzare l'`eks:DescribeCluster`azione per raccogliere informazioni sul cluster a cui si desidera connettere i nodi ibridi. Se non abiliti l'`eks:DescribeCluster`azione, devi passare l'endpoint dell'API Kubernetes, il bundle CA del cluster e il CIDR IPv4 del servizio nella configurazione del nodo che passi al comando. `nodeadm init`
- Autorizzazioni per utilizzare l'`eks:ListAccessEntries`azione per `nodeadm` elencare le voci di accesso sul cluster a cui si desidera connettere i nodi ibridi. Se non abiliti l'`eks:ListAccessEntries`azione, devi passare il `--skip cluster-access-validation` flag quando esegui il `nodeadm init` comando.
- Autorizzazioni per il kubelet per utilizzare le immagini dei contenitori da Amazon Elastic Container Registry (Amazon ECR) come definito nella policy. AmazonEC2ContainerRegistryPullOnly
- Se si utilizza AWS SSM, le autorizzazioni per `nodeadm init` l'utilizzo delle attivazioni ibride AWS SSM sono definite nella policy. https://docs.aws.amazon.com/aws-managed-policy/latest/reference/AmazonSSMManagedInstanceCore.html
- Se si utilizza AWS SSM, le autorizzazioni per utilizzare l'azione e l'`ssm:DeregisterManagedInstance`azione per annullare la registrazione delle istanze`ssm:DescribeInstanceInformation`. `nodeadm uninstall`
- (Facoltativo) Autorizzazioni per consentire al Pod Identity Agent di Amazon EKS di utilizzare l'azione `eks-auth:AssumeRoleForPodIdentity` per recuperare le credenziali per i pod.

Configurazione AWS Attivazioni ibride SSM

**Versione node-adm richiesta per il provider di credenziali SSM**

Se si utilizza AWS Systems Manager (SSM) come provider di credenziali per i nodi ibridi, è necessario utilizzare la `nodeadm` versione `1.0.19` o successiva per le nuove installazioni e gli aggiornamenti. Le versioni precedenti di `nodeadm` contengono una chiave di firma SSM obsoleta e hanno esito negativo durante `nodeadm install` e `nodeadm upgrade` con il seguente errore di verifica della firma:

```
"msg":"Command failed","error":"failed to install ssm installer: validating ssm-setup-cli signature: Signature Verification Error: No matching signature"
```

Per risolvere questo errore, scarica la versione più recente di `nodeadm` prima di eseguire `nodeadm install` o. `nodeadm upgrade`

**Important**

Systems Manager restituisce immediatamente il codice e l'ID di attivazione alla console o finestra di comando, a seconda di come è stato creata l'attivazione. Copia queste informazioni e archiviale in un luogo sicuro. Se esci dalla console o chiudi la finestra di comando, potresti perdere queste informazioni. Se le smarrisci, devi creare una nuova attivazione.

Prima di configurare le attivazioni ibride AWS SSM, è necessario creare e configurare un ruolo IAM Hybrid Nodes. Per ulteriori informazioni, consulta the section called "Creazione del ruolo IAM dei nodi ibridi". Segui le istruzioni in Creare un'attivazione ibrida per registrare i nodi con Systems Manager nella Guida per l'utente di AWS Systems Manager per creare un'attivazione ibrida AWS SSM per i tuoi nodi ibridi. Usa il codice di attivazione e l'ID con `nodeadm` quando registri i tuoi host come nodi ibridi nel tuo cluster Amazon EKS. Puoi tornare a questo passaggio in un secondo momento dopo aver creato e preparato i cluster Amazon EKS per i nodi ibridi.

Per impostazione predefinita, le attivazioni ibride AWS SSM sono attive per 24 ore. In alternativa, puoi specificare una `--expiration-date` quando crei l'attivazione ibrida in formato timestamp, ad esempio `2024-08-01T00:00:00`. Quando utilizzi AWS SSM come provider di credenziali, il nome del nodo per i tuoi nodi ibridi non è configurabile e viene generato automaticamente da SSM. AWS

È possibile visualizzare e gestire le istanze gestite AWS SSM nella console di Systems Manager in Fleet Manager. AWS È possibile registrare fino a 1.000 nodi ibridi standard per account per AWS regione senza costi aggiuntivi. Tuttavia, per registrare più di 1.000 nodi ibridi è necessario attivare il piano istanze avanzate. Viene addebitato un costo per l'utilizzo del piano istanze avanzate che non è incluso nel prezzo di Amazon EKS Hybrid Nodes. Per ulteriori informazioni, consulta Prezzi di AWS Systems Manager.

Guarda l'esempio seguente per come creare un'attivazione ibrida AWS SSM con il tuo ruolo IAM di Hybrid Nodes. Quando utilizzi le attivazioni ibride AWS SSM per le credenziali dei tuoi nodi ibridi, i nomi dei tuoi nodi ibridi hanno il formato `mi-012345678abcdefgh` e le credenziali temporanee fornite da AWS SSM sono valide per 1 ora. Non è possibile modificare il nome del nodo o la durata delle credenziali quando si utilizza SSM come fornitore di credenziali. AWS Le credenziali temporanee vengono ruotate automaticamente da AWS SSM e la rotazione non influisce sullo stato dei nodi o delle applicazioni.

Ti consigliamo di utilizzare un'attivazione ibrida AWS SSM per cluster EKS per limitare l'`ssm:DeregisterManagedInstance`autorizzazione AWS SSM del ruolo IAM di Hybrid Nodes e poter annullare la registrazione solo delle istanze associate all'attivazione ibrida SSM. AWS Nell'esempio in questa pagina, viene utilizzato un tag con l'ARN del cluster EKS, che può essere utilizzato per mappare l'attivazione ibrida AWS SSM sul cluster EKS. In alternativa, puoi utilizzare il tag e il metodo preferiti per definire l'ambito delle autorizzazioni AWS SSM in base ai limiti e ai requisiti di autorizzazione. L'`REGISTRATION_LIMIT`opzione nel comando seguente è un numero intero utilizzato per limitare il numero di macchine che possono utilizzare l'attivazione ibrida AWS SSM (ad esempio) `10`

```
aws ssm create-activation \
     --region AWS_REGION \
     --default-instance-name eks-hybrid-nodes \
     --description "Activation for EKS hybrid nodes" \
     --iam-role AmazonEKSHybridNodesRole \
     --tags Key=EKSClusterARN,Value=arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/
CLUSTER_NAME \
     --registration-limit REGISTRATION_LIMIT
```

Consulta le istruzioni su Crea un'attivazione ibrida per registrare i nodi con Systems Manager per ulteriori informazioni sulle impostazioni di configurazione disponibili per le attivazioni ibride AWS SSM.

### Configurazione AWS IAM Roles Anywhere

Segui le istruzioni riportate in Getting started with IAM Roles Anywhere nella Guida per l'utente di IAM Roles Anywhere per configurare l'ancora di fiducia e il profilo che utilizzerai per le credenziali IAM temporanee per il tuo ruolo IAM dei nodi ibridi. Quando crei il tuo profilo, puoi crearlo senza aggiungere alcun ruolo. Puoi creare questo profilo, tornare a questi passaggi per creare il tuo ruolo IAM dei nodi ibridi e quindi aggiungere il tuo ruolo al profilo dopo averlo creato. In alternativa, puoi utilizzare i AWS CloudFormation passaggi più avanti in questa pagina per completare la configurazione di IAM Roles Anywhere per i nodi ibridi.

Quando aggiungi il ruolo IAM Hybrid Nodes al tuo profilo, seleziona Accetta il nome della sessione del ruolo personalizzato nel pannello Nome della sessione del ruolo personalizzato nella parte inferiore della pagina Modifica profilo nella console AWS IAM Roles Anywhere. Ciò corrisponde al RoleSessionName campo accept dell'`CreateProfile`API. Ciò consente di fornire un nome di nodo personalizzato per i nodi ibridi nella configurazione a cui si passa durante il processo di bootstrap `nodeadm`. È necessario passare un nome di nodo personalizzato durante il processo `nodeadm init`. Puoi aggiornare il tuo profilo per accettare un nome di sessione di ruolo personalizzato dopo aver creato il tuo profilo.

Puoi configurare la durata della validità delle credenziali con AWS IAM Roles Anywhere tramite il campo durationSeconds del tuo profilo AWS IAM Roles Anywhere. La durata predefinita è di 1 ora con un massimo di 12 ore. L'`MaxSessionDuration`impostazione sul ruolo IAM di Hybrid Nodes deve essere maggiore dell'`durationSeconds`impostazione sul profilo AWS IAM Roles Anywhere. Per ulteriori informazioni su`MaxSessionDuration`, consulta la documentazione UpdateRole delle API.

I certificati e le chiavi per computer generati dalla tua autorità di certificazione (CA) devono essere inseriti nella directory `/etc/iam/pki` di ogni nodo ibrido con i nomi dei file `server.pem` per il certificato e `server.key` per la chiave.

### Creazione del ruolo IAM dei nodi ibridi

Per eseguire i passaggi descritti in questa sezione, il responsabile IAM che utilizza la AWS console o la AWS CLI deve disporre delle seguenti autorizzazioni.

- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:AttachRolePolicy`
- Se si utilizza AWS IAM Roles Anywhere

- `rolesanywhere:CreateTrustAnchor`
- `rolesanywhere:CreateProfile`
- `iam:PassRole`

AWS CloudFormation

Installa e configura la AWS CLI, se non l'hai già fatto. Vedi Installazione o aggiornamento alla versione più recente della AWS CLI.

Passaggi per le attivazioni AWS ibride SSM

Lo CloudFormation stack crea il ruolo IAM di Hybrid Nodes con le autorizzazioni descritte sopra. Il CloudFormation modello non crea l'attivazione ibrida AWS SSM.

1. Scarica il CloudFormation modello AWS SSM per i nodi ibridi:

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ssm-cfn.yaml'
```

2. Creare un `cfn-ssm-parameters.json` con le seguenti opzioni:
   a. Sostituisci `ROLE_NAME` con il nome del ruolo IAM dei nodi ibridi. Per impostazione predefinita, il CloudFormation modello utilizza `AmazonEKSHybridNodesRole` come nome del ruolo creato se non si specifica un nome.
   b. Sostituiscilo `TAG_KEY` con la chiave del tag di risorsa AWS SSM che hai utilizzato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della chiave del tag e del valore del tag viene utilizzata nella condizione per consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate all'attivazione ibrida SSM. AWS Nel modello, il valore predefinito è. CloudFormation `TAG_KEY` `EKSClusterARN`
   c. Sostituiscilo `TAG_VALUE` con il valore del tag di risorsa AWS SSM che hai utilizzato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della chiave del tag e del valore del tag viene utilizzata nella condizione per consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate all'attivazione ibrida SSM. AWS Se utilizzi l'impostazione predefinita `TAG_KEY` di `EKSClusterARN`, passa l'ARN del cluster EKS come `TAG_VALUE`. Gli ARN del cluster EKS hanno il formato `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "SSMDeregisterConditionTagKey": "TAG_KEY",
    "SSMDeregisterConditionTagValue": "TAG_VALUE"
  }
}
```

3. Distribuisci lo stack. CloudFormation Sostituiscilo `STACK_NAME` con il tuo nome per lo CloudFormation stack.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ssm-cfn.yaml \
    --parameter-overrides file://cfn-ssm-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

Passaggi per AWS IAM Roles Anywhere

Lo CloudFormation stack crea l'ancoraggio di fiducia AWS IAM Roles Anywhere con l'autorità di certificazione (CA) configurata, crea il profilo AWS IAM Roles Anywhere e crea il ruolo IAM Hybrid Nodes con le autorizzazioni descritte in precedenza.

1. Configurazione di un'autorità di certificazione (CA)
   a. Per utilizzare una risorsa CA AWS privata, apri la console AWS Private Certificate Authority. Segui le istruzioni contenute in AWS Private CA User Guide.
   b. Per utilizzare una CA esterna, segui le istruzioni fornite dalla CA. Fornisci l'ente di certificazione in una fase successiva.
   c. I certificati emessi da CA pubbliche non possono essere utilizzati come ancoraggi di fiducia.
2. Scarica il CloudFormation modello AWS IAM Roles Anywhere per i nodi ibridi

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ira-cfn.yaml'
```

3. Creare un `cfn-iamra-parameters.json` con le seguenti opzioni:

a. Sostituisci `ROLE_NAME` con il nome del ruolo IAM dei nodi ibridi. Per impostazione predefinita, il CloudFormation modello utilizza `AmazonEKSHybridNodesRole` come nome del ruolo creato se non si specifica un nome.

b. Sostituisci `CERT_ATTRIBUTE` con l'attributo del certificato per computer che identifica in modo univoco l'host. L'attributo del certificato utilizzato deve corrispondere al nodeName utilizzato per la configurazione `nodeadm` quando si connettono i nodi ibridi al cluster. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi". Per impostazione predefinita, il CloudFormation modello utilizza `${aws:PrincipalTag/x509Subject/CN}` come`CERT_ATTRIBUTE`, che corrisponde al campo CN dei certificati per macchina. In alternativa puoi passare `$(aws:PrincipalTag/x509SAN/Name/CN}` come tuo `CERT_ATTRIBUTE`.

c. Sostituisci `CA_CERT_BODY` con l'ente del certificato della tua CA senza interruzioni di riga. Il `CA_CERT_BODY` deve essere in formato Privacy-Enhanced Mail (PEM). Se disponi di un certificato CA in formato PEM, rimuovi le interruzioni di riga e le righe BEGIN CERTIFICATE e END CERTIFICATE prima di inserire l'ente del certificato CA nel file `cfn-iamra-parameters.json`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "CertAttributeTrustPolicy": "CERT_ATTRIBUTE",
    "CABundleCert": "CA_CERT_BODY"
  }
}
```

4. Distribuisci il modello. CloudFormation Sostituiscilo `STACK_NAME` con il tuo nome per lo CloudFormation stack.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ira-cfn.yaml \
    --parameter-overrides file://cfn-iamra-parameters.json
    --capabilities CAPABILITY_NAMED_IAM
```

AWS CLI

Installa e configura la AWS CLI, se non l'hai già fatto. Vedi Installazione o aggiornamento alla versione più recente della AWS CLI.

Crea la policy del cluster per descrivere EKS

1. Crea un file denominato `eks-describe-cluster-policy.json` con i seguenti contenuti:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Creare la policy con il seguente comando:

```
aws iam create-policy \
    --policy-name EKSDescribeClusterPolicy \
    --policy-document file://eks-describe-cluster-policy.json
```

Passaggi per le attivazioni AWS ibride SSM

1. Crea un file denominato `eks-hybrid-ssm-policy.json` con i seguenti contenuti. La policy concede l'autorizzazione per le azioni `ssm:DescribeInstanceInformation` e `ssm:DeregisterManagedInstance`. La policy limita l'`ssm:DeregisterManagedInstance`autorizzazione alle istanze gestite da AWS SSM associate all'attivazione ibrida AWS SSM in base al tag di risorsa specificato nella politica di fiducia.
   a. `AWS_REGION`Sostituiscila con la AWS Regione per l'attivazione ibrida SSM AWS .
   b. Sostituisci `AWS_ACCOUNT_ID` con l'ID AWS del tuo account.
   c. Sostituiscila `TAG_KEY` con la chiave tag di risorsa AWS SSM che hai usato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della chiave del tag e del valore del tag viene utilizzata nella condizione per consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate

all'attivazione ibrida SSM. AWS Nel modello, il valore predefinito è. CloudFormation `TAG_KEY` `EKSClusterARN`

d. Sostituiscilo `TAG_VALUE` con il valore del tag di risorsa AWS SSM che hai utilizzato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della chiave del tag e del valore del tag viene utilizzata nella condizione per consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate all'attivazione ibrida SSM. AWS Se utilizzi l'impostazione predefinita `TAG_KEY` di `EKSClusterARN`, passa l'ARN del cluster EKS come `TAG_VALUE`. Gli ARN del cluster EKS hanno il formato `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws:ssm:us-east-1:123456789012:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

2. Creare la policy con il seguente comando

```
aws iam create-policy \
    --policy-name EKSHybridSSMPolicy \
    --policy-document file://eks-hybrid-ssm-policy.json
```

3. Crea un file denominato `eks-hybrid-ssm-trust.json`. `AWS_REGION`Sostituiscila con la AWS regione di attivazione ibrida AWS SSM e con l'ID del tuo account. `AWS_ACCOUNT_ID` AWS

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:ssm:us-east-1:123456789012:*"
            }
         }
      }
   ]
}
```

4. Crea il ruolo con il comando seguente.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-ssm-trust.json
```

5. Collega `EKSDescribeClusterPolicy` e `EKSHybridSSMPolicy` che hai creato nei passaggi precedenti. Sostituiscilo `AWS_ACCOUNT_ID` con l'ID AWS del tuo account.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSHybridSSMPolicy
```

6. Allega `AmazonEC2ContainerRegistryPullOnly` le policy `AmazonSSMManagedInstanceCore` AWS gestite.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
```

Passaggi per AWS IAM Roles Anywhere

Per utilizzare AWS IAM Roles Anywhere, devi configurare il tuo trust anchor AWS IAM Roles Anywhere prima di creare il ruolo IAM Hybrid Nodes. Per istruzioni, consulta the section called "Configurazione AWS IAM Roles Anywhere".

1. Crea un file denominato `eks-hybrid-iamra-trust.json`. Sostituisci `TRUST_ANCHOR ARN` con l'ARN dell'ancora di fiducia che hai creato nei passaggi the section called "Configurazione AWS IAM Roles Anywhere". La condizione di questa policy di fiducia limita la capacità di AWS IAM Roles Anywhere di assumere il ruolo IAM di Hybrid Nodes di scambiare credenziali IAM temporanee solo quando il nome della sessione del ruolo corrisponde al CN nel certificato x509 installato sui nodi ibridi. In alternativa, puoi utilizzare altri attributi del certificato per identificare in modo univoco il tuo nodo. L'attributo del certificato che usi nella policy di fiducia deve corrispondere a `nodeName` che hai impostato nella configurazione `nodeadm`. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}",
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        }
    ]
}
```

2. Crea il ruolo con il comando seguente.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-iamra-trust.json
```

3. Collega la `EKSDescribeClusterPolicy` che hai creato nei passaggi precedenti. Sostituiscilo `AWS_ACCOUNT_ID` con l'ID del tuo account. AWS

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

4. Allega la policy `AmazonEC2ContainerRegistryPullOnly` AWS gestita

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Console di gestione AWS

Crea la policy del cluster per descrivere EKS

1. Apri la console Amazon IAM
2. Nel pannello di navigazione a sinistra, seleziona Policy.
3. Nella pagina Policy, scegli Crea policy.
4. Nella pagina Specifica le autorizzazioni, in Seleziona un pannello di servizio scegli EKS.
   a. Filtra le azioni DescribeCluster e seleziona l'azione DescribeCluster Leggi.
   b. Scegli Next (Successivo).
5. Nella pagina Verifica e crea
   a. Immetti un Nome policy per la tua policy, come `EKSDescribeClusterPolicy`.
   b. Scegli Crea policy.

Passaggi per le attivazioni ibride AWS SSM

1. Apri la console Amazon IAM
2. Nel pannello di navigazione a sinistra, seleziona Policy.
3. Nella pagina Policy, scegli Crea policy.
4. Nella pagina Specifica autorizzazioni, nella barra di navigazione in alto a destra Editor della policy, scegli JSON. Incollare il frammento seguente. Sostituisci `AWS_REGION` con la AWS regione di attivazione ibrida AWS SSM e sostituisci `AWS_ACCOUNT_ID` con l'ID del tuo AWS account. Sostituisci `TAG_KEY` e `TAG_VALUE` con la chiave del tag di risorsa AWS SSM che hai utilizzato durante la creazione dell'attivazione ibrida AWS SSM.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws:ssm:us-east-1:123456789012:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

a. Scegli Next (Successivo).

5. Nella pagina Verifica e crea
   a. Immetti un nome Policy per la tua policy, come `EKSHybridSSMPolicy`.
   b. Scegli Crea policy.
6. Nel pannello di navigazione a sinistra, seleziona Ruoli.
7. Nella pagina Ruoli, seleziona Crea ruolo.
8. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Nel sezione Tipo di entità attendibile, scegli Policy di attendibilità personalizzata. Incolla quanto segue nell'editor della policy di attendibilità personalizzata. `AWS_REGION`Sostituiscila con la AWS regione di attivazione ibrida AWS SSM e `AWS_ACCOUNT_ID` con l'ID del tuo AWS account.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
```

```
                "aws:SourceAccount":"123456789012"
            },
            "ArnEquals":{
                "aws:SourceArn":"arn:aws:ssm:us-east-1:123456789012:*"
            }
         }
      }
   ]
}
```

b. Scegli Next (Successivo).

9. Nella pagina Aggiungi autorizzazioni, collega una policy personalizzata o esegui le operazioni seguenti:

   a. Nella casella Filtra policy, inserisci `EKSDescribeClusterPolicy` o il nome della policy creata in precedenza. Seleziona la casella di controllo a sinistra il nome della policy nei risultati della ricerca.

   b. Nella casella Filtra policy, inserisci `EKSHybridSSMPolicy` o il nome della policy creata in precedenza. Seleziona la casella di controllo a sinistra il nome della policy nei risultati della ricerca.

   c. Nella casella Filtra policy, inserisci `AmazonEC2ContainerRegistryPullOnly`. Seleziona la casella di controllo a sinistra di `AmazonEC2ContainerRegistryPullOnly` nei risultati di ricerca.

   d. Nella casella Filtra policy, inserisci `AmazonSSMManagedInstanceCore`. Seleziona la casella di controllo a sinistra della `AmazonSSMManagedInstanceCore` nei risultati di ricerca.

   e. Scegli Successivo.

10. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:

    a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSHybridNodesRole`.

    b. Per Description (Descrizione), sostituisci il testo corrente con un testo descrittivo come `Amazon EKS - Hybrid Nodes role`.

    c. Scegli Crea ruolo.

Passaggi per AWS IAM Roles Anywhere

Per utilizzare AWS IAM Roles Anywhere, devi configurare il tuo trust anchor AWS IAM Roles Anywhere prima di creare il ruolo IAM Hybrid Nodes. Per istruzioni, consulta the section called "Configurazione AWS IAM Roles Anywhere".

1. Apri la console Amazon IAM
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Nel sezione Tipo di entità attendibile, scegli Policy di attendibilità personalizzata. Incolla quanto segue nell'editor della policy di attendibilità personalizzata. Sostituisci `TRUST_ANCHOR ARN` con l'ARN dell'ancora di fiducia che hai creato nei passaggi the section called "Configurazione AWS IAM Roles Anywhere". La condizione di questa policy di fiducia limita la capacità di AWS IAM Roles Anywhere di assumere il ruolo IAM di Hybrid Nodes di scambiare credenziali IAM temporanee solo quando il nome della sessione del ruolo corrisponde al CN nel certificato x509 installato sui nodi ibridi. In alternativa, puoi utilizzare altri attributi del certificato per identificare in modo univoco il tuo nodo. L'attributo del certificato che usi nella policy di fiducia deve corrispondere al nodeName che hai impostato nella configurazione bideadm. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}",
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        }
    ]
}
```

b. Scegli Next (Successivo).

5. Nella pagina Aggiungi autorizzazioni, collega una policy personalizzata o esegui le operazioni seguenti:

   a. Nella casella Filtra policy, inserisci `EKSDescribeClusterPolicy` o il nome della policy creata in precedenza. Seleziona la casella di controllo a sinistra il nome della policy nei risultati della ricerca.

   b. Nella casella Filtra policy, inserisci `AmazonEC2ContainerRegistryPullOnly`. Seleziona la casella di controllo a sinistra della `AmazonEC2ContainerRegistryPullOnly` nei risultati di ricerca.

   c. Scegli Successivo.

6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:

   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSHybridNodesRole`.

b. Per Description (Descrizione), sostituisci il testo corrente con un testo descrittivo come `Amazon EKS - Hybrid Nodes role`.

c. Scegli Crea ruolo.

## Creazione di un cluster Amazon EKS con nodi ibridi

Questo argomento offre una panoramica delle opzioni disponibili e descrive gli aspetti da considerare quando crei un cluster Amazon EKS abilitato per i nodi ibridi. EKS Hybrid Nodes ha lo stesso supporto per le versioni di Kubernetes dei cluster Amazon EKS con nodi cloud, incluso il supporto standard ed esteso.

Se non hai intenzione di utilizzare EKS Hybrid Nodes, consulta la documentazione principale di Amazon EKS create cluster all'indirizzo the section called "Creazione di un cluster ".

### Prerequisiti

- Il the section called "Prerequisiti" è completato. Prima di creare un cluster abilitato ai nodi ibridi, è necessario che vengano identificati il nodo on-premises e, facoltativamente, i CIDR pod, il VPC e le sottoreti creati in base ai requisiti EKS e ai requisiti dei nodi ibridi e il gruppo di sicurezza con regole in entrata per i tuoi CIDR on-premises, facoltativamente, i pod CIDR. Per ulteriori informazioni su questi prerequisiti, consulta the section called "Preparazione della rete".
- La versione più recente dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. I gestori di pacchetti come yum, apt-get o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione o aggiornamento alla versione più recente della AWS CLI e Configurazione delle impostazioni per la CLI nella Guida per l'utente dell' AWS interfaccia a riga di AWS comando.
- Un principale IAM con le autorizzazioni per creare ruoli IAM e allegare policy, nonché creare e descrivere cluster EKS

### Considerazioni

- Il cluster deve utilizzare una modalità di autenticazione del cluster tra `API` e `API_AND_CONFIG_MAP`.
- Il cluster deve utilizzare la famiglia di indirizzi IPv4.

- Il cluster deve utilizzare la connettività degli endpoint del cluster pubblico o privato. Il tuo cluster non può utilizzare la connettività endpoint del cluster "pubblica e privata", poiché l'endpoint del server API Amazon EKS Kubernetes si risolverà negli IP pubblici per i nodi ibridi in esecuzione al di fuori del tuo VPC.
- L'autenticazione OIDC è supportata per i cluster EKS con nodi ibridi.
- È possibile aggiungere, modificare o rimuovere la configurazione dei nodi ibridi dei cluster esistenti. Per ulteriori informazioni, consulta the section called "Cluster esistente".
- Se non specifichi esplicitamente un servizio Kubernetes IPv4 CIDR durante la creazione di un cluster ibrido abilitato ai nodi, Amazon EKS seleziona automaticamente un servizio CIDR che non si sovrappone alle reti di nodi e pod remoti configurati. Il CIDR assegnato può differire dai valori predefiniti standard (o). `10.100.0.0/16` `172.20.0.0/16` Ad esempio, se le reti di nodi o pod remoti si sovrappongono a questi intervalli predefiniti, Amazon EKS seleziona un CIDR alternativo dagli intervalli di IP privati come. `172.16.0.0/16` Ti consigliamo di specificare esplicitamente un servizio CIDR IPv4 durante la creazione del cluster per garantire una configurazione prevedibile. È possibile trovare il servizio assegnato CIDR dopo la creazione del cluster eseguendo. `aws eks describe-cluster --name CLUSTER_NAME --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"`

Passaggio 1: creazione di un ruolo IAM del cluster

Se disponi già di un ruolo IAM del cluster o intendi creare il tuo cluster con `eksctl` o AWS CloudFormation, puoi saltare questo passaggio. Per impostazione predefinita, `eksctl` il AWS CloudFormation modello crea automaticamente il ruolo IAM del cluster.

1. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Crea il ruolo IAM del cluster Amazon EKS. Se necessario, inserisci il prefisso eks-cluster-role-trust-policy.json nel percorso del computer in cui hai scritto il file nella Passaggio precedente. Il comando associa il criterio di attendibilità creato nella Passaggio precedente al ruolo. Per creare un ruolo IAM, l'azione `iam:CreateRole` (autorizzazione) deve essere assegnata al principale IAM che sta creando il ruolo.

```
aws iam create-role \
    --role-name myAmazonEKSClusterRole \
    --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

3. Puoi assegnare la policy gestita da Amazon EKS o creare una policy personalizzata. Per conoscere le autorizzazioni minime che devi utilizzare nella policy personalizzata, consulta the section called "Ruolo IAM del nodo". Allega al ruolo la policy gestita da Amazon EKS, denominata `AmazonEKSClusterPolicy`. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

```
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
    --role-name myAmazonEKSClusterRole
```

Passaggio 2: creazione di cluster abilitato per i nodi ibridi

Puoi creare un cluster utilizzando:

- eksctl
- AWS CloudFormation
- AWS CLI
- Console di gestione AWS

Crea un cluster abilitato per i nodi ibridi - eksctl

È necessario installare la versione più recente dello strumento della riga di comando `eksctl`. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

1. Crea `cluster-config.yaml` per definire un cluster Amazon EKS IPv4 abilitato ai nodi ibridi. Effettua le seguenti sostituzioni in `cluster-config.yaml`. Per un elenco completo delle impostazioni, consulta eksctl documentation.

   a. Sostituisci `CLUSTER_NAME` con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.

   b. Sostituiscilo `AWS_REGION` con la AWS regione in cui desideri creare il cluster.

   c. Sostituisci `K8S_VERSION` con qualsiasi versione supportata da Amazon EKS.

   d. Sostituisci `CREDS_PROVIDER` con `ssm` o `ira` in base al provider di credenziali che hai configurato nei passaggi per the section called "Preparazione delle credenziali".

   e. Sostituisci `CA_BUNDLE_CERT` se il tuo provider di credenziali è impostato su`ira`, che utilizza AWS IAM Roles Anywhere come provider di credenziali. Il CA_BUNDLE_CERT è l'ente di certificazione dell'autorità di certificazione (CA) e dipende dalla CA scelta. Il certificato deve essere in formato Privacy-Enhanced Mail (PEM).

   f. Sostituisci `GATEWAY_ID` con l'ID del tuo gateway privato virtuale o gateway di transito da collegare al tuo VPC.

   g. Sostituisci `REMOTE_NODE_CIDRS` con il nodo on-premises CIDR per i tuoi nodi ibridi.

   h. Sostituisci `REMOTE_POD_CIDRS` con il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi o rimuovi la riga dalla configurazione se non esegui webhook su nodi ibridi. È necessario configurare `REMOTE_POD_CIDRS` se il CNI non utilizza Network Address Translation (NAT) o il mascheramento per gli indirizzi IP dei pod quando il traffico dei pod lascia gli host on-premises. È necessario eseguire la configurazione `REMOTE_POD_CIDRS` se si eseguono webhook su nodi ibridi; per ulteriori informazioni consulta the section called "Configurazione di webhook".

   i. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:

      i. Rientrare in uno degli RFC-1918 intervalli IPv4:`10.0.0.0/8`, `172.16.0.0/12192.168.0.0/16`, o o all'interno dell'intervallo CGNAT definito da RFC 6598:. `100.64.0.0/10`

      ii. Non si sovrappongano tra loro, né con `VPC  CIDR` per il tuo cluster né con il tuo servizio Kubernetes (IPv4 CIDR)

```
apiVersion: eksctl.io/v1alpha5
```

```
kind: ClusterConfig

metadata:
  name: CLUSTER_NAME
  region: AWS_REGION
  version: "K8S_VERSION"

remoteNetworkConfig:
  iam:
    provider: CREDS_PROVIDER # default SSM, can also be set to IRA
    # caBundleCert: CA_BUNDLE_CERT
  vpcGatewayID: GATEWAY_ID
  remoteNodeNetworks:
  - cidrs: ["REMOTE_NODE_CIDRS"]
  remotePodNetworks:
  - cidrs: ["REMOTE_POD_CIDRS"]
```

2. Esegui il comando seguente:

```
eksctl create cluster -f cluster-config.yaml
```

Il provisioning del cluster richiede diversi minuti. Durante la creazione del cluster, vengono visualizzate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

```
[#]  EKS cluster "CLUSTER_NAME" in "REGION" region is ready
```

3. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Crea un cluster ibrido abilitato ai nodi - AWS CloudFormation

Lo CloudFormation stack crea il ruolo IAM del cluster EKS e un cluster EKS con `RemoteNodeNetwork` e `RemotePodNetwork` da te specificato. Modifica il CloudFormation modello Se devi personalizzare le impostazioni per il tuo cluster EKS che non sono esposte nel CloudFormation modello.

1. Scaricate il CloudFormation modello.

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-eks-cfn.yaml'
```

2. Crea `cfn-eks-parameters.json` e specifica la tua configurazione per ogni valore.

a. `CLUSTER_NAME`: il nome del nuovo cluster EKS da creare.

b. `CLUSTER_ROLE_NAME`: il nome del nuovo ruolo IAM del cluster EKS da creare. L'impostazione predefinita nel modello è «EKSClusterRole».

c. `SUBNET1_ID`: l'ID della prima sottorete creata nei passaggi preliminari.

d. `SUBNET2_ID`: l'ID della seconda sottorete creata nei passaggi preliminari.

e. `SG_ID`: il gruppo di sicurezza creato nei passaggi preliminari.

f. `REMOTE_NODE_CIDRS`: il nodo on-premises CIDR per i tuoi nodi ibridi.

g. `REMOTE_POD_CIDRS`: il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi. È necessario configurare `REMOTE_POD_CIDRS` se il CNI non utilizza Network Address Translation (NAT) o il mascheramento per gli indirizzi IP dei pod quando il traffico dei pod lascia gli host on-premises. È necessario eseguire la configurazione `REMOTE_POD_CIDRS` se si eseguono webhook su nodi ibridi; per ulteriori informazioni consulta the section called "Configurazione di webhook".

h. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:

   i. Rientrare in uno degli RFC-1918 intervalli IPv4:`10.0.0.0/8`, `172.16.0.0/12192.168.0.0/16`, o o all'interno dell'intervallo CGNAT definito da RFC 6598:. `100.64.0.0/10`

   ii. Non si sovrappongano tra loro, né con `VPC  CIDR` per il tuo cluster né con il servizio Kubernetes IPv4 CIDR.

i. `CLUSTER_AUTH`: la modalità di autenticazione del cluster per il tuo cluster. I valori validi sono `API` e `API_AND_CONFIG_MAP`. L'impostazione predefinita nel modello è `API_AND_CONFIG_MAP`.

j. `CLUSTER_ENDPOINT`: la connettività degli endpoint del cluster per il cluster. I valori validi sono "Public" e "Private". L'impostazione predefinita nel modello è Private, il che significa che potrai connetterti all'endpoint dell'API Kubernetes solo dall'interno del tuo VPC.

k. `K8S_VERSION`: la versione Kubernetes desiderata per il cluster. Consulta Versioni supportate da Amazon EKS.

```
{
  "Parameters": {
    "ClusterName": "CLUSTER_NAME",
    "ClusterRoleName": "CLUSTER_ROLE_NAME",
    "SubnetId1": "SUBNET1_ID",
    "SubnetId2": "SUBNET2_ID",
    "SecurityGroupId" "SG_ID",
```

```
    "RemoteNodeCIDR": "REMOTE_NODE_CIDRS",
    "RemotePodCIDR": "REMOTE_POD_CIDRS",
    "ClusterAuthMode": "CLUSTER_AUTH",
    "ClusterEndpointConnectivity": "CLUSTER_ENDPOINT",
    "K8sVersion": "K8S_VERSION"
  }
 }
```

3. Implementa lo stack. CloudFormation Sostituiscilo `STACK_NAME` con il tuo nome per CloudFormation lo stack e `AWS_REGION` con la AWS regione in cui verrà creato il cluster.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --template-file hybrid-eks-cfn.yaml \
    --parameter-overrides file://cfn-eks-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

   Il provisioning del cluster richiede diversi minuti. È possibile controllare lo stato dello stack con il comando seguente. Sostituiscilo `STACK_NAME` con il tuo nome per CloudFormation lo stack e `AWS_REGION` con la AWS regione in cui verrà creato il cluster.

```
aws cloudformation describe-stacks \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --query 'Stacks[].StackStatus'
```

4. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Crea un cluster ibrido abilitato ai nodi - AWS CLI

1. Esegui il comando seguente per creare un cluster EKS abilitato per i nodi ibridi. Prima di eseguire il comando, sostituisci le impostazioni seguenti con le tue. Per un elenco completo delle impostazioni, consulta la documentazione the section called "Creazione di un cluster ".
   a. `CLUSTER_NAME`: il nome del nuovo cluster EKS da creare.
   b. `AWS_REGION`: AWS regione in cui verrà creato il cluster.
   c. `K8S_VERSION`: la versione Kubernetes desiderata per il cluster. Controlla le versioni supportate da Amazon EKS.

d. ROLE_ARN: il ruolo del cluster Amazon EKS che hai configurato per il tuo cluster. Per ulteriori informazioni, consulta il ruolo IAM del cluster Amazon EKS.

e. SUBNET1_ID: l'ID della prima sottorete creata nei passaggi preliminari.

f. SUBNET2_ID: l'ID della seconda sottorete creata nei passaggi preliminari.

g. SG_ID: il gruppo di sicurezza creato nei passaggi preliminari.

h. È possibile utilizzare API e API_AND_CONFIG_MAP per la modalità di autenticazione dell'accesso al cluster. Nel comando seguente, la modalità di autenticazione dell'accesso al cluster è impostata su API_AND_CONFIG_MAP.

i. È possibile utilizzare i parametri endpointPublicAccess e endpointPrivateAccess per abilitare o disabilitare l'accesso pubblico e privato all'endpoint del server API Kubernetes del cluster. Nel comando seguente endpointPublicAccess è impostato su false e endpointPrivateAccess è impostato su true.

j. REMOTE_NODE_CIDRS: il nodo on-premises CIDR per i tuoi nodi ibridi.

k. REMOTE_POD_CIDRS (facoltativo): il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi.

l. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:

   i. Rientrare in uno degli RFC-1918 intervalli IPv4: 10.0.0.0/8 172.16.0.0/12192.168.0.0/16, o o o all'interno dell'intervallo CGNAT definito da RFC 6598:. 100.64.0.0/10

   ii. Non si sovrappongano tra loro, né con VPC CIDR per il tuo cluster Amazon EKS né con il tuo servizio Kubernetes IPv4 CIDR.

```
aws eks create-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --kubernetes-version K8S_VERSION \
    --role-arn ROLE_ARN \
    --resources-vpc-config
 subnetIds=SUBNET1_ID,SUBNET2_ID,securityGroupIds=SG_ID,endpointPrivateAccess=true,endpo
 \
    --access-config authenticationMode=API_AND_CONFIG_MAP \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. Il provisioning del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente. Sostituisci CLUSTER_NAME con il nome del cluster che stai creando e

`AWS_REGION` con la AWS regione in cui il cluster sta creando. Non passare alla fase successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Crea un cluster abilitato ai nodi ibridi - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Seleziona Aggiungi cluster e quindi Crea.
3. Nella pagina Configura cluster compila i campi seguenti:
   a. Nome: un nome per il cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
   b. Ruolo IAM del cluster: scegli il ruolo IAM del cluster Amazon EKS che hai creato per consentire al piano di controllo Kubernetes di gestire AWS le risorse per tuo conto.
   c. Versione Kubernetes – la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra una versione precedente.
   d. Policy di aggiornamento: scegli tra esteso o standard.
      i. Esteso: questa opzione supporta la versione Kubernetes per 26 mesi dopo la data di rilascio. Il periodo di supporto esteso ha un costo orario aggiuntivo che inizia dopo la fine del periodo di quello standard. Al termine del supporto esteso, il cluster verrà aggiornato automaticamente alla versione successiva.
      ii. Standard: questa opzione supporta la versione Kubernetes per 14 mesi dopo la data di rilascio. Non è previsto alcun costo aggiuntivo Al termine del supporto standard, il cluster verrà aggiornato automaticamente alla versione successiva.
   e. Accesso al cluster: scegli di consentire o negare l'accesso all'amministratore del cluster e seleziona una modalità di autenticazione. Le seguenti modalità di autenticazione sono supportate per i cluster abilitati per i nodi ibridi.
      i. API EKS: il cluster fornirà i principali IAM autenticati solo dalle API di accesso EKS.

ii. API EKS e ConfigMap: Il cluster fornirà i principali IAM autenticati sia dalle API di accesso EKS che da. `aws-auth` ConfigMap

f. Crittografia dei segreti (Facoltativo): scegli di abilitare la crittografia dei segreti per i segreti Kubernetes tramite una chiave KMS. Puoi abilitare questa funzionalità anche dopo la creazione del cluster. Assicurati, tuttavia, di acquisire familiarità con le informazioni contenute nella sezione the section called "Abilitare la crittografia dei segreti".

g. Spostamento zonale ARC: se abilitato, EKS registrerà il cluster con lo spostamento zonale ARC per consentirti di utilizzare quest'ultimo per spostare il traffico delle applicazioni lontano da una zona di disponibilità.

h. Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Tagging delle risorse ".

i. Quando hai finito con questa pagina, seleziona Avanti.

4. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:

a. VPC: scegli un VPC esistente che soddisfi i requisiti di the section called "Requisiti del VPC e delle sottoreti" e Amazon EKS Hybrid Nodes. Prima di scegliere un VPC, ti consigliamo di acquisire familiarità con i requisiti e le considerazioni nella sezione Visualizza i requisiti di rete di Amazon EKS per VPC, sottoreti e nodi ibridi. Dopo la creazione del cluster, non potrai più modificare il VPC da utilizzare. Se l'elenco risulta vuoto, è necessario crearne uno prima. Per ulteriori informazioni consulta the section called "Crea un VPC" e Amazon EKS Hybrid Nodes networking requirements.

b. Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. È necessario selezionarne almeno due.

c. Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS. Almeno uno dei gruppi di sicurezza specificati deve disporre di regole in entrata per il nodo on-premises e, facoltativamente, dei pod CIDR. Per ulteriori informazioni, consulta Amazon EKS Hybrid Nodes networking requirements. Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". È possibile modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.

d. Scegli la famiglia di indirizzi IP del cluster: devi scegliere IPv4 per i cluster abilitati per i nodi ibridi.

e. (Facoltativo) Scegli Configura l'intervallo di indirizzi IP del servizio Kubernetes e specifica un intervallo del servizio IPv4. Consigliamo di specificare un servizio CIDR esplicito per i cluster abilitati ai nodi ibridi. Se non ne specifichi uno, Amazon EKS seleziona un CIDR che evita i conflitti con le reti di nodi e pod remoti, che possono differire dai valori predefiniti standard (o). `10.100.0.0/16` `172.20.0.0/16`

f. Scegli Configura reti remote per abilitare i nodi ibridi e specifica i CIDR di nodi e pod on-premises per i nodi ibridi.

g. È necessario configurare i pod CIDR remoti se il CNI non utilizza Network Address Translation (NAT) o il mascheramento per gli indirizzi IP dei pod quando il traffico dei pod lascia gli host on-premises. È necessario configurare il pod CIDR remoto se si eseguono webhook su nodi ibridi.

h. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:

   i. Rientra in uno degli RFC-1918 intervalli IPv4: `10.0.0.0/8`172.16.0.0/12, o o rientra nell'intervallo CGNAT `192.168.0.0/16` definito da RFC 6598:. `100.64.0.0/10`

   ii. Non si sovrappongano tra loro, né con `VPC CIDR` per il tuo cluster né con il tuo servizio Kubernetes (IPv4 CIDR)

i. Per Cluster endpoint access (Accesso all'endpoint del cluster), seleziona un'opzione. Puoi modificare questa opzione dopo aver creato il cluster. Per i cluster abilitati per i nodi ibridi, devi scegliere Pubblico o Privato. Prima di selezionare un'opzione non predefinita, assicurati di familiarizzare con le opzioni e le relative implicazioni. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

j. Quando hai finito con questa pagina, seleziona Avanti.

5. (Facoltativo) Nella pagina Configura osservabilità, scegli le opzioni Parametri e Registrazione del piano di controllo da attivare. Per impostazione predefinita, i tipi di log sono disattivati.

   a. Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Parametri Prometheus".

   b. Per ulteriori informazioni sulle opzioni Registrazione del controllo EKS, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

   c. Quando hai finito con questa pagina, seleziona Avanti.

6. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster.

   a. Puoi scegliere tutti i componenti aggiuntivi Amazon EKS e i componenti aggiuntivi AWS Marketplace di cui hai bisogno. I componenti aggiuntivi di Amazon EKS che non sono compatibili con i nodi ibridi sono contrassegnati con la dicitura "Non compatibile con nodi ibridi" e hanno una regola di anti-affinità per impedirne l'esecuzione su nodi ibridi. Per

ulteriori informazioni, consulta Configurazione dei componenti aggiuntivi per i nodi ibridi. Se i componenti aggiuntivi del  AWS Marketplace che desideri installare non sono elencati, puoi cercare i componenti aggiuntivi di AWS Marketplace disponibili inserendo il testo nella casella di ricerca. Puoi eseguire una ricerca anche per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca.

b. Alcuni componenti aggiuntivi, come CoreDNS e kube-proxy, sono installati per impostazione predefinita. Disabilitare uno dei componenti aggiuntivi predefiniti potrebbe influire sulla tua capacità di eseguire le applicazioni Kubernetes.

c. Quando hai finito con questa pagina, seleziona `Next`.

7. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona la versione da installare.

   a. Dopo la creazione del cluster, puoi eseguire in qualsiasi momento l'aggiornamento a una versione successiva. Puoi aggiornare la configurazione di ogni componente aggiuntivo dopo la creazione del cluster. Per ulteriori informazioni sulla configurazione dei componenti aggiuntivi, consulta the section called "Aggiornamento di un componente aggiuntivo". Per le versioni dei componenti aggiuntivi compatibili con i nodi ibridi, consulta the section called "Configurare i componenti aggiuntivi".

   b. Quando hai finito con questa pagina, seleziona Avanti.

8. Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE. Il provisioning del cluster richiede diversi minuti.

9. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Passaggio 3: aggiornamento di kubeconfig

Se hai creato il cluster utilizzando `eksctl`, puoi ignorare questo passaggio, poiché è già stato completato da `eksctl`. Abilita `kubectl` per consentire la comunicazione con il cluster aggiungendo un nuovo contesto al file di configurazione `kubectl`. Per ulteriori informazioni su come creare e aggiornare il file, consulta the section called "Accedere al cluster con kubectl".

```
aws eks update-kubeconfig --name CLUSTER_NAME --region AWS_REGION
```

Di seguito viene riportato un output di esempio:

```
Added new context arn:aws: eks:AWS_REGION:111122223333:cluster/CLUSTER_NAME to /home/
username/.kube/config
```

Conferma la comunicazione con il cluster eseguendo il comando seguente.

```
kubectl get svc
```

Di seguito viene riportato un output di esempio.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Passaggio 4: configurazione del cluster

Come passaggio successivo, consulta the section called "Preparazione dell'accesso ai cluster" per abilitare l'accesso ai nodi ibridi al fine di entrare a far parte del cluster.

## Abilitazione dei nodi ibridi su un cluster Amazon EKS esistente o modifica della configurazione

Questo argomento offre una panoramica delle opzioni disponibili e descrive gli aspetti da considerare quando aggiungi, modifichi o rimuovi la configurazione dei nodi ibridi per un cluster Amazon EKS.

Per consentire a un cluster Amazon EKS di utilizzare nodi ibridi, aggiungi gli intervalli CIDR degli indirizzi IP del nodo on-premises e, facoltativamente, la rete pod nella configurazione `RemoteNetworkConfig`. EKS utilizza questo elenco di CIDR per abilitare la connettività tra il cluster e le reti on-premises. Per un elenco completo delle opzioni per l'aggiornamento della configurazione del UpdateClusterConfigcluster, consulta Amazon EKS API Reference.

Puoi eseguire una delle seguenti azioni sulla configurazione della rete dei nodi ibridi EKS in un cluster:

- Aggiungi una configurazione di rete remota per abilitare i nodi ibridi EKS in un cluster esistente.
- Aggiungi, modifica o rimuovi le reti di nodi remoti e quelle pod remote in un cluster esistente.
- Rimuovi tutti gli intervalli CIDR della rete di nodi remoti per disabilitare i nodi ibridi EKS in un cluster esistente.

## Prerequisiti

- Prima di abilitare il cluster Amazon EKS per nodi ibridi, assicurati che l'ambiente soddisfi i requisiti descritti in the section called "Prerequisiti" e specificati nel dettaglio in the section called "Preparazione della rete", the section called "Preparazione del sistema operativo" e the section called "Preparazione delle credenziali".
- Il cluster deve utilizzare la famiglia di indirizzi IPv4.
- Il cluster deve utilizzare una modalità di autenticazione del cluster tra `API` e `API_AND_CONFIG_MAP`. Il processo di modifica della modalità di autenticazione del cluster è descritto in the section called "Modalità di autenticazione".
- Ti consigliamo di utilizzare l'accesso all'endpoint pubblico o privato per l'endpoint del server API Amazon EKS Kubernetes, ma non entrambi. Se scegli "Pubblico e privato", l'endpoint del server API Amazon EKS Kubernetes si risolverà sempre negli IP pubblici per i nodi ibridi in esecuzione all'esterno del tuo VPC, il che può impedire ai nodi ibridi di unirsi al cluster. Il processo per modificare l'accesso di rete al cluster è descritto in the section called "Accesso all'endpoint del cluster".
- La versione più recente dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. I gestori di pacchetti come yum, apt-get o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione o aggiornamento alla versione più recente della AWS CLI e Configurazione delle impostazioni per la CLI nella Guida per l'utente dell' AWS interfaccia a riga di AWS comando.
- Un principale IAM con l'autorizzazione a chiamare UpdateClusterConfigil tuo cluster Amazon EKS.
- Aggiorna i componenti aggiuntivi alle versioni compatibili con i nodi ibridi. Per le versioni dei componenti aggiuntivi compatibili con i nodi ibridi, consulta the section called "Configurare i componenti aggiuntivi".
- Se esegui componenti aggiuntivi non compatibili con i nodi ibridi, assicurati che il componente aggiuntivo DaemonSeto la distribuzione abbiano la seguente regola di affinità per impedire la distribuzione su nodi ibridi. Aggiungi la seguente regola di affinità se non è già presente.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
```

```
          values:
          - hybrid
```

Considerazioni

L'oggetto `remoteNetworkConfig` JSON ha il seguente comportamento durante un aggiornamento:

- Qualsiasi parte esistente della configurazione che non specifichi rimane invariata. Se non specificate nessuno né `remoteNodeNetworks` né `remotePodNetworks`, quella parte rimarrà la stessa.
- Se modifichi l'elenco `remoteNodeNetworks` o l'elenco `remotePodNetworks` dei CIDR, è necessario specificare quello completo dei CIDR che si desidera inserire nella configurazione finale. Quando specifichi una modifica all'elenco `remoteNodeNetworks` o all'elenco `remotePodNetworks` CIDR, EKS sostituisce l'elenco originale durante l'aggiornamento.
- I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:
  1. Rientrare in uno degli intervalli RFC-1918 IPv4:10.0.0. 0/8, 172.16.0. 0/12, o 192.168.0. 0/16 o all'interno dell'intervallo CGNAT definito dalla RFC 6598: `100.64.0.0/10`
  2. Non si sovrappongano tra loro, né con tutti i CIDR del VPC per il tuo cluster Amazon EKS né con il tuo servizio Kubernetes IPv4 CIDR.

Abilitazione dei nodi ibridi su un cluster esistente

È possibile abilitare i nodi ibridi EKS in un cluster esistente utilizzando:

- AWS CloudFormation
- AWS CLI
- Console di gestione AWS

Abilita i nodi ibridi EKS in un cluster esistente - AWS CloudFormation

1. Per abilitare i nodi ibridi EKS nel tuo cluster, aggiungi `RemoteNodeNetwork` e (opzionale) `RemotePodNetwork` al CloudFormation modello e aggiorna lo stack. Tieni presente che `RemoteNodeNetwork` è un elenco con un massimo di un elemento `Cidrs` e che `Cidrs` è un elenco di più intervalli IP CIDR.

```
RemoteNetworkConfig:
```

```
  RemoteNodeNetworks:
    - Cidrs: [RemoteNodeCIDR]
  RemotePodNetworks:
    - Cidrs: [RemotePodCIDR]
```

2. Continua su the section called "Preparazione dell'accesso ai cluster".

Abilita i nodi ibridi EKS in un cluster esistente - AWS CLI

1. Esegui il comando seguente per abilitare `RemoteNetworkConfig` per i nodi ibridi EKS per il cluster EKS. Prima di eseguire il comando, sostituisci le impostazioni seguenti con le tue. Per un elenco completo delle impostazioni, consulta UpdateClusterConfigAmazon EKS API Reference.
   a. `CLUSTER_NAME`: nome del cluster EKS da aggiornare
   b. `AWS_REGION`: AWS Regione in cui è in esecuzione il cluster EKS.
   c. `REMOTE_NODE_CIDRS`: il nodo on-premises CIDR per i tuoi nodi ibridi.
   d. `REMOTE_POD_CIDRS` (facoltativo): il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. L'aggiornamento del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente. Sostituisci `CLUSTER_NAME` con il nome del cluster che stai modificando e `AWS_REGION` con la AWS regione in cui è in esecuzione il cluster. Non passare alla fase successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Continua su the section called "Preparazione dell'accesso ai cluster".

Abilita i nodi ibridi EKS in un cluster esistente - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.

2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegliere la scheda Reti, quindi Gestione.
4. Nel menu a discesa, scegli Reti remote.
5. Scegli Configura reti remote per abilitare i nodi ibridi e specifica i CIDR di nodi e pod on-premises per i nodi ibridi.
6. Scegliere Salva le modifiche per terminare. Attendi che lo stato del cluster ritorni su Attivo.
7. Continua su the section called "Preparazione dell'accesso ai cluster".

### Aggiornamento della configurazione dei nodi ibridi in un cluster esistente

È possibile effettuare modifiche `remoteNetworkConfig` in un cluster ibrido esistente utilizzando uno dei seguenti metodi:

- AWS CloudFormation
- AWS CLI
- Console di gestione AWS

#### Aggiorna la configurazione ibrida in un cluster esistente - AWS CloudFormation

1. Aggiorna il CloudFormation modello con i nuovi valori CIDR di rete.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [NEW_REMOTE_NODE_CIDRS]
  RemotePodNetworks:
    - Cidrs: [NEW_REMOTE_POD_CIDRS]
```

> **Note**
>
> Quando aggiorni gli elenchi `RemoteNodeNetworks` o `RemotePodNetworks` CIDR, includi tutti i CIDR (nuovi ed esistenti). EKS sostituisce l'intero elenco durante gli aggiornamenti. L'omissione di questi campi dalla richiesta di aggiornamento mantiene le configurazioni esistenti.

2. Aggiorna CloudFormation lo stack con il modello modificato e attendi il completamento dell'aggiornamento dello stack.

Aggiorna la configurazione ibrida in un cluster esistente - AWS CLI

1. Per modificare i CIDR della rete remota, eseguire questo comando. Sostituisci i valori le tue impostazioni:

```
aws eks update-cluster-config
--name CLUSTER_NAME
--region AWS_REGION
--remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["NEW_REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":
["NEW_REMOTE_POD_CIDRS"]}]}'
```

> **Note**
>
> Quando aggiorni gli elenchi `remoteNodeNetworks` o `remotePodNetworks` CIDR, includi tutti i CIDR (nuovi ed esistenti). EKS sostituisce l'intero elenco durante gli aggiornamenti. L'omissione di questi campi dalla richiesta di aggiornamento mantiene le configurazioni esistenti.

2. Attendi che lo stato del cluster ritorni ad ATTIVO prima di procedere.

Aggiorna la configurazione ibrida in un cluster esistente - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegliere la scheda Reti, quindi Gestione.
4. Nel menu a discesa, scegli Reti remote.
5. Aggiorna i CIDR sotto `Remote node networks` e `Remote pod networks - Optional` secondo necessità.
6. Scegli Salva modifiche e attendi che lo stato del cluster ritorni Attivo.

Disabilitazione dei nodi ibridi in un cluster esistente

È possibile disabilitare i nodi ibridi EKS in un cluster esistente utilizzando:

- AWS CloudFormation
- AWS CLI

- Console di gestione AWS

Disabilita i nodi ibridi EKS in un cluster esistente - AWS CloudFormation

1. Per disabilitare i nodi ibridi EKS nel cluster, imposta `RemoteNodeNetworks` e `RemotePodNetworks` svuota gli array nel CloudFormation modello e aggiorna lo stack.

```
RemoteNetworkConfig:
  RemoteNodeNetworks: []
  RemotePodNetworks: []
```

Disabilita i nodi ibridi EKS in un cluster esistente - AWS CLI

1. Esegui i seguenti comandi per rimuovere `RemoteNetworkConfig` dal cluster EKS. Prima di eseguire il comando, sostituisci le impostazioni seguenti con le tue. Per un elenco completo delle impostazioni, consulta UpdateClusterConfigAmazon EKS API Reference.
   a. `CLUSTER_NAME`: nome del cluster EKS da aggiornare
   b. `AWS_REGION`: AWS Regione in cui è in esecuzione il cluster EKS.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[],"remotePodNetworks":[]}'
```

2. L'aggiornamento del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente. Sostituisci `CLUSTER_NAME` con il nome del cluster che stai modificando e `AWS_REGION` con la AWS regione in cui è in esecuzione il cluster. Non passare alla fase successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

Disabilita i nodi ibridi EKS in un cluster esistente - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.

2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegliere la scheda Reti, quindi Gestione.
4. Nel menu a discesa, scegli Reti remote.
5. Scegli Configura reti remote per abilitare i nodi ibridi e rimuovi tutti i CIDR sotto `Remote node networks` e `Remote pod networks - Optional`.
6. Scegliere Salva le modifiche per terminare. Attendi che lo stato del cluster ritorni su Attivo.

## Preparazione dell'accesso al cluster per i nodi ibridi

Prima di connettere i nodi ibridi al cluster Amazon EKS, devi abilitare le autorizzazioni del ruolo IAM per i nodi ibridi con Kubernetes per entrare a far parte del cluster. Per ulteriori informazioni su come creare il ruolo IAM per i nodi ibridi, consulta the section called "Preparazione delle credenziali". Amazon EKS supporta due modi per associare i principali IAM a Kubernetes Role-Based Access Control (RBAC), alle voci di accesso Amazon EKS e a. `aws-auth` ConfigMap Per ulteriori informazioni sulla gestione dell'accesso di Amazon EKS, consultathe section called "Accesso API Kubernetes" .

Utilizza le procedure seguenti per associare il ruolo IAM dei nodi ibridi alle autorizzazioni Kubernetes. Per utilizzare le voci di accesso di Amazon EKS, il cluster deve essere stato creato con le modalità di autenticazione `API` o `API_AND_CONFIG_MAP`. Per utilizzare il `aws-auth` ConfigMap, il cluster deve essere stato creato con la modalità di autenticazione. `API_AND_CONFIG_MAP` La modalità di autenticazione `CONFIG_MAP`-only non è supportata per i cluster Amazon EKS abilitati per i nodi ibridi.

Utilizzo delle voci di accesso di Amazon EKS per il ruolo IAM dei nodi ibridi

Esiste un tipo di accesso Amazon EKS per i nodi ibridi denominato HYBRID_LINUX che può essere utilizzato con un ruolo IAM. Con questo tipo di accesso, il nome utente viene impostato automaticamente su system:node: {{SessionName}}. Per ulteriori informazioni sulla creazione delle voci di accesso, consulta the section called "Creare voci di accesso".

AWS CLI

1. È necessario che sul dispositivo sia installata e configurata la versione più recente della AWS CLI. Per verificare la versione attuale, usa `aws --version`. I gestori di pacchetti come yum, apt-get o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell' AWS interfaccia a riga di comando.

2. Creare la tua voce di accesso con il comando seguente. Sostituisci CLUSTER_NAME con il nome del tuo cluster e HYBRID_NODES_ROLE_ARN con l'ARN del ruolo che hai creato nei passaggi per the section called "Preparazione delle credenziali".

```
aws eks create-access-entry --cluster-name CLUSTER_NAME \
    --principal-arn HYBRID_NODES_ROLE_ARN \
    --type HYBRID_LINUX
```

Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Scegli il nome del cluster ibrido abilitato per i nodi.
3. Scegliere la scheda Accesso.
4. Seleziona Crea voce di accesso.
5. Per principale IAM, seleziona il ruolo IAM dei nodi ibridi che hai creato nei passaggi per the section called "Preparazione delle credenziali".
6. Per Tipo, seleziona Hybrid Linux.
7. (Facoltativo) Per Tag, assegna etichette alla voce di accesso. Ad esempio, per facilitare la ricerca di tutte le risorse con lo stesso tag.
8. Scegli Vai alla revisione e alla creazione. Non è possibile aggiungere policy alla voce di accesso Hybrid Linux o modificarne l'ambito di accesso.
9. Verifica la configurazione per la tua voce di accesso. Se noti qualcosa di errato, scegli Precedente per tornare indietro e correggere l'errore. Se la configurazione è corretta, scegli Crea.

Utilizzo di aws-auth ConfigMap per il ruolo IAM di Hybrid Nodes

Nei passaggi seguenti, creerai o aggiornerai `aws-auth` ConfigMap con l'ARN il ruolo IAM dei nodi ibridi che hai creato nei passaggi per. the section called "Preparazione delle credenziali"

1. Verifica se ne hai uno esistente `aws-auth` ConfigMap per il tuo cluster. Considera che se stai usando un file `kubeconfig` specifico, devi usare il flag `--kubeconfig`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrato un `aws-auth` ConfigMap, aggiornalo secondo necessità.

a. Apri il file ConfigMap per modificarlo.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Aggiungi una nuova voce `mapRoles`, se necessario. Sostituisci `HYBRID_NODES_ROLE_ARN` con l'ARN del ruolo IAM dei nodi ibridi. Nota, `{{SessionName}}` è il formato di modello corretto da salvare in ConfigMap. Non sostituirlo con altri valori.

```
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
```

c. Salva il file ed esci dall'editor di testo.

3. Se non ne esiste uno `aws-auth` ConfigMap per il tuo cluster, crealo con il seguente comando. Sostituisci `HYBRID_NODES_ROLE_ARN` con l'ARN del ruolo IAM dei nodi ibridi. Nota che `{{SessionName}}` è il formato di modello corretto da salvare in ConfigMap. Non sostituirlo con altri valori.

```
kubectl apply -f=/dev/stdin <<-EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
EOF
```

# Esegui carichi di lavoro on-premises su nodi ibridi

In un cluster EKS con nodi ibridi abilitati, puoi eseguire applicazioni on-premises ed edge sulla tua infrastruttura con gli stessi cluster, le stesse funzionalità e gli stessi strumenti Amazon EKS che usi nel cloud AWS.

Le seguenti sezioni contengono istruzioni dettagliate per l'utilizzo di nodi ibridi.

Argomenti


- Connessione di nodi ibridi
- Connessione dei nodi ibridi con Bottlerocket
- Aggiornamento dei nodi ibridi per il tuo cluster
- Aggiornamenti di sicurezza delle patch per i nodi ibridi
- Rimuovi i nodi ibridi


## Connessione di nodi ibridi

> **Note**
>
> I seguenti passaggi si applicano ai nodi ibridi che eseguono sistemi operativi compatibili ad eccezione di Bottlerocket. Per i passaggi per connettere un nodo ibrido che esegue Bottlerocket, consultare the section called "Connessione dei nodi ibridi con Bottlerocket".

In questo argomento viene descritto come connettere nodi ibridi a un cluster Amazon EKS. Una volta entrati nel cluster, i nodi ibridi appariranno con lo stato Non pronti nella console Amazon EKS e in Kubernetes-compatible strumenti come kubectl. Dopo aver completato i passaggi in questa pagina, procedi su the section called "Configurazione della CNI" per preparare i nodi ibridi per l'esecuzione delle applicazioni.

Prerequisiti

Prima di connettere i nodi ibridi al cluster Amazon EKS, assicurati di aver completato tutti i passaggi relativi ai prerequisiti.

- Disponi della connettività di rete dal tuo ambiente locale alla AWS regione che ospita il tuo cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Preparazione della rete".

- Si dispone di un sistema operativo compatibile per i nodi ibridi installato sugli host on-premises. Per ulteriori informazioni, consulta the section called "Preparazione del sistema operativo".
- Hai creato il tuo ruolo IAM Hybrid Nodes e impostato il tuo provider di credenziali locale (attivazioni ibride di AWS Systems Manager o AWS IAM Roles Anywhere). Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".
- Hai creato il cluster Amazon EKS abilitato per i nodi ibridi. Per ulteriori informazioni, consulta the section called "Creazione di un cluster".
- Hai associato il tuo ruolo IAM di Hybrid Nodes alle autorizzazioni Kubernetes Role-Based Access Control (RBAC). Per ulteriori informazioni, consulta the section called "Preparazione dell'accesso ai cluster".

⚠ Versione nodeadm richiesta per il provider di credenziali SSM

Se si utilizza AWS Systems Manager (SSM) come provider di credenziali per i nodi ibridi, è necessario utilizzare la `nodeadm` versione `1.0.19` o successiva per le nuove installazioni e gli aggiornamenti. Le versioni precedenti di `nodeadm` contengono una chiave di firma SSM obsoleta e hanno esito negativo durante `nodeadm install` e `nodeadm upgrade` con il seguente errore di verifica della firma:

```
"msg":"Command failed","error":"failed to install ssm installer: validating ssm-setup-cli signature: Signature Verification Error: No matching signature"
```

Per risolvere questo errore, scarica la versione più recente di `nodeadm` prima di eseguire `nodeadm install` o. `nodeadm upgrade`

Passaggio 1: installare la CLI dei nodi ibridi (**`nodeadm`**) su ogni host locale

Se si include la CLI Amazon EKS Hybrid Nodes (`nodeadm`) nelle immagini predefinite del sistema operativo, è possibile saltare questo passaggio. Per maggiori informazioni sulla versione per nodi ibridi del `nodeadm`, consultare the section called "nodeadm dei nodi ibridi".

La versione dei nodi ibridi di `nodeadm` è ospitata in Amazon S3 gestita da Amazon. CloudFront Per installare `nodeadm` su ciascun host on-premises, puoi eseguire il seguente comando dagli host on-premises.

Per host x86_64:

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

Per host ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

Aggiungi l'autorizzazione per i file eseguibili al file binario scaricato su ciascun host.

```
chmod +x nodeadm
```

Passaggio 2: installa le dipendenze dei nodi ibridi con nodeadm

Se si stanno installando le dipendenze dei nodi ibridi in immagini del sistema operativo predefinite, è possibile ignorare questo passaggio. Il comando `nodeadm install` può essere utilizzato per installare tutte le dipendenze richieste per i nodi ibridi. Le dipendenze dei nodi ibridi includono i componenti containerd, kubelet, kubectl e SSM o IAM Roles Anywhere. AWS AWS Consultare the section called "nodeadm dei nodi ibridi" per ulteriori informazioni sui componenti e le posizioni dei file installati da `nodeadm install`. Consultare la sezione the section called "Preparazione della rete" dedicata ai nodi ibridi per ulteriori informazioni sui domini che devono essere consentiti nel firewall on-premises per il processo `nodeadm install`.

Eseguire il comando seguente per installare le dipendenze dei nodi ibridi sull'host on-premises. Il comando seguente deve essere eseguito con un utente che ha accesso al tuo host. sudo/root

> **Important**
>
> La CLI (`nodeadm`) dei nodi ibridi deve essere eseguita con un utente che ha sudo/root accesso sull'host.

- Sostituire `K8S_VERSION` con la versione secondaria Kubernetes del cluster Amazon EKS, ad esempio `1.31`. Consultare Amazon EKS supported versions per un elenco delle versioni Kubernetes supportate.
- Sostituire `CREDS_PROVIDER` con il provider di credenziali on-premises che si sta utilizzando. I valori validi sono `ssm` per AWS SSM e `iam-ra` per AWS IAM Roles Anywhere.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER
```

Passaggio 3: connessione dei nodi ibridi al cluster

Prima di connettere i nodi ibridi al cluster, assicurati di aver consentito l'accesso richiesto nel firewall locale e nel gruppo di sicurezza del cluster per la comunicazione tra nodi to/from ibridi del piano di controllo Amazon EKS. La maggior parte dei problemi in questa Passaggio riguarda la configurazione del firewall, la configurazione del gruppo di sicurezza o la configurazione del ruolo IAM dei nodi ibridi.

**Important**

La CLI (`nodeadm`) dei nodi ibridi deve essere eseguita con un utente che ha sudo/root accesso sul tuo host.

1. Crea un file `nodeConfig.yaml` su ogni host con i valori per la tua implementazione. Per una descrizione completa delle impostazioni di configurazione disponibili, consulta the section called "nodeadm dei nodi ibridi". Se il ruolo IAM dei nodi ibridi non dispone dell'autorizzazione per l'azione `eks:DescribeCluster`, devi passare l'endpoint dell'API Kubernetes, il cluster CA bundle e il servizio Kubernetes IPv4 CIDR nella sezione cluster del tuo `nodeConfig.yaml`.
   a. Utilizza l'`nodeConfig.yaml`esempio seguente se utilizzi attivazioni ibride AWS SSM per il tuo provider di credenziali locale.
      i. Sostituisci `CLUSTER_NAME` con il nome del cluster.
      ii. Sostituiscilo `AWS_REGION` con la AWS regione che ospita il tuo cluster. Ad esempio, `us-west-2`.
      iii. Sostituiscilo `ACTIVATION_CODE` con il codice di attivazione che hai ricevuto durante la creazione dell'attivazione ibrida AWS SSM. Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".
      iv. Sostituiscilo `ACTIVATION_ID` con l'ID di attivazione che hai ricevuto durante la creazione dell'attivazione ibrida AWS SSM. È possibile recuperare queste informazioni dalla console di AWS Systems Manager o dal comando AWS `aws ssm describe-activations` CLI.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
```

```
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    ssm:
      activationCode: ACTIVATION_CODE
      activationId: ACTIVATION_ID
```

b. Usa l'`nodeConfig.yaml`esempio seguente se stai usando AWS IAM Roles Anywhere per il tuo provider di credenziali locale.

   i. Sostituisci `CLUSTER_NAME` con il nome del cluster.

   ii. Sostituiscilo `AWS_REGION` con la AWS regione che ospita il tuo cluster. Ad esempio, `us-west-2`.

   iii. Sostituisci `NODE_NAME` con il nome del ruolo del nodo. Il nome del nodo deve corrispondere al CN del certificato sull'host se hai configurato la policy di attendibilità del ruolo IAM di Hybrid Nodes con la condizione della risorsa `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`. Il `nodeName` che usi non deve superare i 64 caratteri.

   iv. Sostituisci `TRUST_ANCHOR_ARN` con l'ARN dell'ancora di fiducia che hai configurato nei passaggi per Preparare le credenziali per i nodi ibridi.

   v. Sostituisci `PROFILE_ARN` con l'ARN dell'ancora di fiducia che hai configurato nei passaggi per the section called "Preparazione delle credenziali".

   vi. Sostituisci `ROLE_ARN` con l'ARN del ruolo IAM dei nodi ibridi.

   vii. Sostituisci `CERTIFICATE_PATH` con il percorso su disco del certificato del tuo nodo. Se non lo specifichi, il valore predefinito è `/etc/iam/pki/server.pem`.

   viii. Sostituisci `KEY_PATH` con il percorso su disco della chiave privata del certificato. Se non lo specifichi, il valore predefinito è `/etc/iam/pki/server.key`.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    iamRolesAnywhere:
      nodeName: NODE_NAME
      trustAnchorArn: TRUST_ANCHOR_ARN
      profileArn: PROFILE_ARN
```

```
    roleArn: ROLE_ARN
    certificatePath: CERTIFICATE_PATH
    privateKeyPath: KEY_PATH
```

2. Esegui il comando `nodeadm init` con il tuo `nodeConfig.yaml` per connettere i nodi ibridi al cluster Amazon EKS.

```
nodeadm init -c file://nodeConfig.yaml
```

Se il comando precedente viene effettuato correttamente, il nodo ibrido si unisce al cluster Amazon EKS. Puoi verificarlo nella console Amazon EKS accedendo alla scheda Calcolo del tuo cluster (assicurati che il principale IAM disponga delle autorizzazioni per la visualizzazione) o con `kubectl get nodes`.

> **Important**
>
> I nodi avranno lo stato `Not Ready` previsto, dovuto alla mancanza di una CNI in esecuzione sui nodi ibridi. Se i tuoi nodi non si sono uniti al cluster, consulta the section called "Risoluzione dei problemi".

Passaggio 4: configurazione di una CNI per i nodi ibridi

Per preparare i nodi ibridi all'esecuzione delle applicazioni, continua con i passaggi su the section called "Configurazione della CNI".

## Connessione dei nodi ibridi con Bottlerocket

Questo argomento descrive come connettere i nodi ibridi che eseguono Bottlerocket a un cluster Amazon EKS. Bottlerocket è una distribuzione Linux open source sponsorizzata e supportata da. AWS Bottlerocket è progettato appositamente per ospitare carichi di lavoro in container. Con Bottlerocket, puoi migliorare la disponibilità delle implementazioni containerizzate e ridurre i costi operativi automatizzando gli aggiornamenti dell'infrastruttura dei container. Bottlerocket include solo il software essenziale per eseguire i container, che migliora l'utilizzo delle risorse, riduce le minacce alla sicurezza e riduce i costi di gestione.

Solo VMware le varianti della versione Bottlerocket v1.37.0 e successive sono supportate con EKS Hybrid Nodes. VMware le varianti di Bottlerocket sono disponibili per le versioni Kubernetes v1.28

e successive. Le immagini del sistema operativo per queste varianti includono kubelet, containerd aws-iam-authenticator e altri prerequisiti software per EKS Hybrid Nodes. È possibile configurare questi componenti utilizzando un file di impostazioni Bottlerocket che include dati utente codificati in base64 per i contenitori bootstrap e admin di Bottlerocket. La configurazione di queste impostazioni consente a Bottlerocket di utilizzare il provider di credenziali dei nodi ibridi per autenticare i nodi ibridi nel cluster. Dopo che i nodi ibridi si sono uniti al cluster, verranno visualizzati con lo stato `Not Ready` nella console Amazon EKS e in strumenti compatibili con Kubernetes come `kubectl`. Dopo aver completato i passaggi indicati in questa pagina, procedi a preparare i nodi ibridi the section called "Configurazione della CNI" per l'esecuzione delle applicazioni.

Prerequisiti

Prima di connettere i nodi ibridi al cluster Amazon EKS, assicurati di aver completato tutti i passaggi relativi ai prerequisiti.

- Hai una connettività di rete dal tuo ambiente locale alla AWS regione che ospita il tuo cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Preparazione della rete".
- Hai creato il tuo ruolo IAM Hybrid Nodes e configurato il tuo provider di credenziali locale (AWS Systems Manager hybrid activations o AWS IAM Roles Anywhere). Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".
- Hai creato il cluster Amazon EKS abilitato per i nodi ibridi. Per ulteriori informazioni, consulta the section called "Creazione di un cluster".
- Hai associato il ruolo IAM Nodi ibridi alle autorizzazioni con il controllo degli accessi basato sui ruoli (RBAC) di Kubernetes. Per ulteriori informazioni, consulta the section called "Preparazione dell'accesso ai cluster".

Passaggio 1: creazione del file TOML delle impostazioni Bottlerocket

Per configurare Bottlerocket per i nodi ibridi, è necessario creare un file `settings.toml` con la configurazione necessaria. Il contenuto del file TOML sarà diverso in base al provider di credenziali utilizzato (SSM o IAM Roles Anywhere). Questo file verrà passato come dati utente durante il provisioning dell'istanza Bottlerocket.

> **Note**
>
> I file TOML forniti di seguito rappresentano solo le impostazioni minime richieste per inizializzare una VMWare macchina Bottlerocket come nodo su un cluster EKS. Bottlerocket

offre un'ampia gamma di impostazioni per soddisfare diversi casi d'uso, quindi per ulteriori opzioni di configurazione oltre all'inizializzazione del nodo ibrido, consulta la documentazione di Bottlerocket per l'elenco completo di tutte le impostazioni documentate per la versione Bottlerocket che stai utilizzando (ad esempio, ecco tutte le impostazioni disponibili per Bottlerocket 1.51.x).

SSM

Se utilizzi AWS Systems Manager come provider di credenziali, crea un `settings.toml` file con il seguente contenuto:

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"
cloud-provider = ""
server-tls-bootstrap = true

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"

[settings.kubernetes.credential-providers.ecr-credential-provider]
enabled = true
cache-duration = "12h"
image-patterns = [
    "*.dkr.ecr.*.amazonaws.com",
    "*.dkr.ecr.*.amazonaws.com.cn",
    "*.dkr.ecr.*.amazonaws.eu",
    "*.dkr.ecr-fips.*.amazonaws.com",
    "*.dkr.ecr-fips.*.amazonaws.eu",
    "public.ecr.aws"
]

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
```

```
"eks.amazonaws.com/hybrid-credential-provider" = "ssm"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"

[settings.host-containers.control]
enabled = true
```

Sostituire i segnaposto con i seguenti valori:

- `<cluster-name>`: il nome del cluster Amazon EKS.
- `<api-server-endpoint>`: l'endpoint del server API del cluster.
- `<cluster-certificate-authority>`: il bundle CA con codifica base64 del cluster.
- `<region>`: La AWS regione che ospita il cluster, ad esempio «us-east-1".
- `<hostname>`: il nome host dell'istanza Bottlerocket, che verrà configurato anche come nome del nodo. Questo può essere un valore univoco a tua scelta, ma deve rispettare le convenzioni di denominazione degli oggetti Kubernetes. Inoltre, il nome host che usi non può superare i 64 caratteri. NOTA: quando si utilizza il provider SSM, il nome host e il nome del nodo verranno sostituiti dall'ID dell'istanza gestita (ad esempio, `mi-*` ID) dopo la registrazione dell'istanza con SSM.
- `<base64-encoded-admin-container-userdata>`: il contenuto con codifica base64 della configurazione del container di amministrazione Bottlerocket. L'abilitazione del container di amministrazione consente di connettersi all'istanza Bottlerocket con SSH per l'esplorazione e il debug del sistema. Sebbene questa non sia un'impostazione obbligatoria, consigliamo di abilitarla per facilitare la risoluzione dei problemi. Consulta Bottlerocket admin container documentation per ulteriori informazioni sull'autenticazione con il container di amministrazione. Il container di amministrazione accetta l'input dell'utente e della chiave SSH in formato JSON, ad esempio

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
```

```
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: il contenuto con codifica base64 della configurazione del container del bootstrap Bottlerocket. Consulta Bottlerocket bootstrap container documentation per ulteriori informazioni sulla sua configurazione. Il contenitore bootstrap è responsabile della registrazione dell'istanza come istanza gestita AWS SSM e dell'aggiunta come nodo Kubernetes sul tuo cluster Amazon EKS. I dati utente passati nel container bootstrap assumono la forma di un'invocazione di comando che accetta come input il codice di attivazione ibrido SSM e l'ID creati in precedenza:

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-code=<activation-
code> --region=<region>
```

IAM Roles Anywhere

Se utilizzi AWS IAM Roles Anywhere come provider di credenziali, crea un file con il seguente contenuto: `settings.toml`

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"
cloud-provider = ""
server-tls-bootstrap = true

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"
config = "<base64-encoded-aws-config-file>"

[settings.kubernetes.credential-providers.ecr-credential-provider]
enabled = true
cache-duration = "12h"
```

```
image-patterns = [
    "*.dkr.ecr.*.amazonaws.com",
    "*.dkr.ecr.*.amazonaws.com.cn",
    "*.dkr.ecr.*.amazonaws.eu",
    "*.dkr.ecr-fips.*.amazonaws.com",
    "*.dkr.ecr-fips.*.amazonaws.eu",
    "public.ecr.aws"
]

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "iam-ra"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"
```

Sostituire i segnaposto con i seguenti valori:

- `<cluster-name>`: il nome del cluster Amazon EKS.
- `<api-server-endpoint>`: l'endpoint del server API del cluster.
- `<cluster-certificate-authority>`: il bundle CA con codifica base64 del cluster.
- `<region>`: La AWS regione che ospita il cluster, ad esempio «us-east-1"
- `<hostname>`: il nome host dell'istanza Bottlerocket, che verrà configurato anche come nome del nodo. Questo può essere un valore univoco a tua scelta, ma deve rispettare le convenzioni di denominazione degli oggetti Kubernetes. Inoltre, il nome host che usi non può superare i 64 caratteri. NOTA: quando si utilizza il provider IAM-RA, il nome del nodo deve corrispondere al CN del certificato sull'host se è stata configurata la policy di attendibilità del ruolo IAM dei nodi ibridi con la condizione della risorsa `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`.
- `<base64-encoded-aws-config-file>`: I contenuti codificati in base64 del file di configurazione. AWS Il contenuto del file dovrebbe essere il seguente:

```
[default]
```

```
credential_process = aws_signing_helper credential-process --certificate /root/.aws/
node.crt --private-key /root/.aws/node.key --profile-arn <profile-arn> --role-arn
 <role-arn> --trust-anchor-arn <trust-anchor-arn> --role-session-name <role-session-
name>
```

- `<base64-encoded-admin-container-userdata>`: il contenuto con codifica base64 della configurazione del container di amministrazione Bottlerocket. L'abilitazione del container di amministrazione consente di connettersi all'istanza Bottlerocket con SSH per l'esplorazione e il debug del sistema. Sebbene questa non sia un'impostazione obbligatoria, consigliamo di abilitarla per facilitare la risoluzione dei problemi. Consulta Bottlerocket admin container documentation per ulteriori informazioni sull'autenticazione con il container di amministrazione. Il container di amministrazione accetta l'input dell'utente e della chiave SSH in formato JSON, ad esempio

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: il contenuto con codifica base64 della configurazione del container del bootstrap Bottlerocket. Consulta Bottlerocket bootstrap container documentation per ulteriori informazioni sulla sua configurazione. Il container bootstrap è responsabile della creazione del certificato host IAM Roles Anywhere e dei file delle chiavi private del certificato sull'istanza. Questi verranno quindi utilizzati da `aws_signing_helper` per ottenere credenziali temporanee per l'autenticazione con il cluster Amazon EKS. I dati utente passati nel container bootstrap assumono la forma di un'invocazione di comando che accetta come input i contenuti del certificato la chiave privata creati in precedenza:

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key>
```

Passaggio 2: fornitura dei dati utente alla macchina virtuale Bottlerocket vSphere

Dopo aver creato il file TOML, passalo come dati utente durante la creazione della macchina virtuale vSphere. Tieni presente che i dati utente devono essere configurati prima che la macchina virtuale

venga accesa per la prima volta. Pertanto, sarà necessario fornirli al momento della creazione dell'istanza o, se si desidera creare la macchina virtuale in anticipo; quest'ultima deve essere nello stato PoweredOff fino a quando non si configurano i relativi dati utente. Ad esempio, se utilizzi la `govc` CLI:

Creazione della macchina virtuale per la prima volta

```
govc vm.create \
  -on=true \
  -c=2 \
  -m=4096 \
  -net.adapter=<network-adapter> \
  -net=<network-name> \
  -e guestinfo.userdata.encoding="base64" \
  -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
  -template=<template-name> \
  <vm-name>
```

Aggiornamento dei dati utente per una macchina virtuale esistente

```
govc vm.create \
    -on=false \
    -c=2 \
    -m=4096 \
    -net.adapter=<network-adapter> \
    -net=<network-name> \
    -template=<template-name> \
    <vm-name>

govc vm.change
    -vm <vm-name> \
    -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
    -e guestinfo.userdata.encoding="base64"

govc vm.power -on <vm-name>
```

Nelle sezioni precedenti, l'opzione `-e guestinfo.userdata.encoding="base64"` specifica che i dati utente sono codificati con Base64. L'opzione `-e guestinfo.userdata` passa i contenuti codificati con Base64 del file `settings.toml` come dati utente all'istanza Bottlerocket. Sostituisci i segnaposto con i tuoi valori specifici, come il modello OVA Bottlerocket e i dettagli di rete.

Passaggio 3: verifica della connessione del nodo ibrido

Dopo l'avvio, l'istanza Bottlerocket tenterà di entrare a far parte del tuo cluster Amazon EKS. Puoi verificare la connessione nella console Amazon EKS accedendo alla scheda Calcolo per il tuo cluster o eseguendo il seguente comando:

```
kubectl get nodes
```

> **Important**
>
> I nodi avranno lo stato `Not Ready` previsto, dovuto alla mancanza di una CNI in esecuzione sui nodi ibridi. Se i tuoi nodi non si sono uniti al cluster, consulta the section called "Risoluzione dei problemi".

Passaggio 4: configurazione di una CNI per i nodi ibridi

Per preparare i nodi ibridi all'esecuzione delle applicazioni, continua con i passaggi su the section called "Configurazione della CNI".

## Aggiornamento dei nodi ibridi per il tuo cluster

Le linee guida per l'aggiornamento dei nodi ibridi sono simili ai nodi Amazon EKS autogestiti eseguiti in Amazon EC2. Ti consigliamo di creare nuovi nodi ibridi nella versione di Kubernetes di destinazione, eseguire la migrazione con facilità delle applicazioni esistenti sui nodi ibridi sulla nuova versione di Kubernetes e rimuovere questi ultimi sulla vecchia versione di Kubernetes dal tuo cluster. Assicurati di consultare le Best practice di Amazon EKS per gli aggiornamenti prima di iniziare un aggiornamento. Amazon EKS Hybrid Nodes ha lo stesso Kubernetes version support per i cluster Amazon EKS con nodi cloud, incluso il supporto standard ed esteso.

Amazon EKS Hybrid Nodes segue la stessa policy di disallineamento delle versioni per i nodi di Kubernetes upstream. Amazon EKS Hybrid Nodes non può avere una versione più recente del piano di controllo di Amazon EKS e i nodi ibridi possono contenere fino a tre versioni secondarie di Kubernetes precedenti a quella secondaria del piano di controllo Amazon EKS.

Se non disponi di capacità di riserva per creare nuovi nodi ibridi sulla versione Kubernetes di destinazione per una strategia di conversione della migrazione, puoi in alternativa utilizzare la CLI di Amazon EKS Hybrid Nodes (`nodeadm`) per aggiornare la versione Kubernetes dei tuoi nodi ibridi sul posto.

**Important**

Se stai aggiornando i tuoi nodi ibridi sul posto con `nodeadm`, è previsto un tempo di inattività per il nodo durante la procedura, in cui la versione precedente dei componenti di Kubernetes viene chiusa e i componenti della nuova versione vengono installati e avviati.

**Versione node-adm richiesta per il provider di credenziali SSM**

Se si utilizza AWS Systems Manager (SSM) come provider di credenziali per i nodi ibridi, è necessario utilizzare la `nodeadm` versione `1.0.19` o successiva per le nuove installazioni e gli aggiornamenti. Le versioni precedenti di `nodeadm` contengono una chiave di firma SSM obsoleta e hanno esito negativo durante `nodeadm install` e `nodeadm upgrade` con il seguente errore di verifica della firma:

```
"msg":"Command failed","error":"failed to install ssm installer: validating ssm-setup-cli signature: Signature Verification Error: No matching signature"
```

Per risolvere questo errore, scarica la versione più recente di `nodeadm` prima di eseguire `nodeadm install` o. `nodeadm upgrade`

## Prerequisiti

Prima di effettuare la connessione, assicurati di avere soddisfatto tutti i prerequisiti.

- La versione Kubernetes di destinazione per l'aggiornamento dei nodi ibridi deve essere uguale o inferiore a quella del piano di controllo di Amazon EKS.
- Se stai seguendo una strategia di conversione della migrazione e dell'aggiornamento, i nuovi nodi ibridi che stai installando sulla versione Kubernetes di destinazione devono soddisfare i requisiti the section called "Prerequisiti". Ciò include la presenza di indirizzi IP all'interno del CIDR della rete di nodi remoti che hai trasferito durante la creazione del cluster Amazon EKS.
- Sia per la conversione della migrazione che per gli aggiornamenti sul posto, i nodi ibridi devono avere accesso ai domini richiesti per sfruttare le nuove versioni delle dipendenze dei nodi ibridi.
- Devi avere kubectl installato sul computer o sull'istanza locale che stai utilizzando per interagire con l'endpoint dell'API Amazon EKS Kubernetes.

- La versione del tuo CNI deve supportare la versione di Kubernetes verso cui stai effettuando l'aggiornamento. In caso contrario, aggiorna la versione del CNI prima di aggiornare i nodi ibridi. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

Aggiornamenti conversione migrazione (blu-verde)

Gli aggiornamenti della conversione della migrazione si riferiscono al processo di creazione di nuovi nodi ibridi su nuovi host con la versione di Kubernetes di destinazione, con la migrazione graduale delle applicazioni esistenti ai nuovi nodi ibridi sulla versione Kubernetes di destinazione e la rimozione dei nodi ibridi sulla vecchia versione di Kubernetes dal cluster. Questa strategia è anche chiamata migrazione blu-verde.

1. Connetti i tuoi nuovi host come nodi ibridi seguendo i passaggi di the section called "Connessione di nodi ibridi". Quando esegui il comando `nodeadm install`, utilizza la versione di Kubernetes di destinazione.
2. Abilita la comunicazione tra i nuovi nodi ibridi nella versione Kubernetes di destinazione e quelli nella vecchia versione di Kubernetes. Ciò consente ai pod di comunicare tra loro durante la migrazione del carico di lavoro verso i nodi ibridi nella versione Kubernetes di destinazione.
3. Verifica che i nodi ibridi sulla versione di Kubernetes di destinazione si uniscano correttamente al cluster e mostrino lo stato Pronto.
4. Utilizza il comando seguente per contrassegnare come non programmabile ciascuno dei nodi che desideri rimuovere. In questo modo i nuovi pod non vengono pianificati né riprogrammati sui nodi che stai sostituendo. Per ulteriori informazioni, consulta cordonazione di kubelet nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome dei nodi ibridi nella vecchia versione di Kubernetes.

   ```
   kubectl cordon NODE_NAME
   ```

   Puoi individuare e isolare tutti i nodi di una determinata versione di Kubernetes (in questo caso `1.28`) con il seguente frammento di codice.

   ```
   K8S_VERSION=1.28
   for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
    "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
    match("\($K8S_VERSION)")).metadata.name')
   do
   ```

```
    echo "Cordoning $node"
    kubectl cordon $node
done
```

5. Se l'implementazione attuale è in esecuzione per un numero di volte inferiore a due repliche CoreDNS, aumenta orizzontalmente l'implementazione a due repliche. Consigliamo di eseguire almeno due repliche CoreDNS su nodi ibridi per garantire la resilienza durante le normali operazioni.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

6. Svuota ciascuno dei nodi della vecchia versione di Kubernetes che desideri rimuovere dal cluster con il comando seguente. Per ulteriori informazioni su come svuotare i nodi, consulta Safely Drain a Node nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome dei nodi ibridi nella vecchia versione di Kubernetes.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

   Puoi individuare e svuotare tutti i nodi di una determinata versione di Kubernetes (in questo caso, `1.28`) con il seguente frammento di codice.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-emptydir-data
done
```

7. Puoi utilizzare `nodeadm` per arrestare e rimuovere gli artefatti dei nodi ibridi dall'host. È necessario eseguire l'`nodeadm`operazione con un utente con root/sudo privilegi. Per impostazione predefinita, `nodeadm uninstall` non proseguirà se sul nodo rimangono dei pod. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
nodeadm uninstall
```

8. Una volta interrotti e disinstallati gli artefatti dei nodi ibridi, rimuovi la risorsa del nodo dal cluster.

```
kubectl delete node node-name
```

Puoi individuare ed eliminare tutti i nodi di una determinata versione di Kubernetes (in questo caso `1.28`) con il seguente frammento di codice.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Deleting $node"
    kubectl delete node $node
done
```

9. A seconda del CNI scelto, potrebbero rimanere degli artefatti sui nodi ibridi dopo aver eseguito i passaggi precedenti. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

In-place aggiornamenti

La procedura di aggiornamento sul posto si riferisce all'utilizzo di `nodeadm upgrade` per l'aggiornamento della versione di Kubernetes per nodi ibridi senza utilizzare nuovi host fisici o virtuali e alla conversione della strategia di migrazione. Il processo `nodeadm upgrade` chiude i componenti Kubernetes precedenti esistenti in esecuzione sul nodo ibrido, disinstalla i componenti Kubernetes precedenti esistenti, installa i nuovi componenti Kubernetes di destinazione e avvia i nuovi componenti Kubernetes di destinazione. Si consiglia vivamente di aggiornare un nodo alla volta per ridurre al minimo l'impatto sulle applicazioni in esecuzione sui nodi ibridi. La durata di questo processo dipende dalla larghezza di banda e dalla latenza della rete.

1. Usa il comando seguente per contrassegnare come non programmabile il nodo che stai aggiornando. In questo modo i nuovi pod non vengono pianificati né riprogrammati sul nodo che stai sostituendo. Per ulteriori informazioni, consulta cordonazione di kubelet nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome del nodo ibrido che stai aggiornando

```
kubectl cordon NODE_NAME
```

2. Svuota il nodo da aggiornare con il comando seguente. Per ulteriori informazioni su come svuotare i nodi, consulta Safely Drain a Node nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome del nodo ibrido che stai aggiornando.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

3. Esegui `nodeadm upgrade` sul nodo ibrido che stai aggiornando. È necessario eseguire l'esecuzione `nodeadm` con un utente con root/sudo privilegi. Il nome del nodo viene preservato tramite l'aggiornamento dei provider di credenziali AWS SSM e AWS IAM Roles Anywhere. Durante il processo di aggiornamento non puoi modificare i provider di credenziali. Consulta the section called "nodeadm dei nodi ibridi" per i valori di configurazione per `nodeConfig.yaml`. Sostituisci `K8S_VERSION` con la versione di Kubernetes di destinazione verso cui esegui l'aggiornamento.

```
nodeadm upgrade K8S_VERSION -c file://nodeConfig.yaml
```

4. Per consentire la pianificazione dei pod sul nodo dopo l'aggiornamento, digita quanto segue. Sostituisci `NODE_NAME` con il nome del nodo.

```
kubectl uncordon NODE_NAME
```

5. Osserva lo stato dei nodi ibridi e attendi fino a quando i nodi si chiudono e si riavviano sulla nuova versione di Kubernetes con lo stato Pronto.

```
kubectl get nodes -o wide -w
```

## Aggiornamenti di sicurezza delle patch per i nodi ibridi

Questo argomento descrive la procedura per eseguire l'applicazione sul posto delle patch degli aggiornamenti di sicurezza per pacchetti e dipendenze specifici in esecuzione sui nodi ibridi. Come best practice, ti consigliamo di aggiornare regolarmente i nodi ibridi per ricevere CVE e patch di sicurezza.

Per i passaggi per aggiornare la versione di Kubernetes, consulta the section called "Aggiornamento dei nodi ibridi".

Un esempio di software che potrebbe richiedere l'applicazione di patch di sicurezza è `containerd`.

### `Containerd`

`containerd` è il runtime standard del container Kubernetes e la dipendenza principale per EKS Hybrid Nodes. Viene utilizzato per gestire il ciclo di vita dei container, inclusa l'estrazione delle

immagini e la gestione dell'esecuzione dei container. Su un nodo ibrido, è possibile eseguire l'installazione `containerd` tramite la CLI nodeadm o manualmente. A seconda del sistema operativo del nodo, `nodeadm` verrà installato `containerd` dal OS-distributed pacchetto o dal pacchetto Docker.

Una volta pubblicato un CVE in `containerd`, hai le seguenti opzioni per eseguire l'aggiornamento alla versione con patch di `containerd` sui tuoi nodi ibridi.

Passaggio 1: controlla se la patch è stata pubblicata sui gestori di pacchetti

Puoi verificare se la patch CVE `containerd` è stata pubblicata su ogni rispettivo gestore di pacchetti del sistema operativo facendo riferimento ai corrispondenti bollettini sulla sicurezza:

- Amazon Linux 2023
- RHEL
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

Se utilizzi il repository Docker come fonte di `containerd`, puoi controllare gli annunci di sicurezza del Docker per identificare la disponibilità della versione con patch nel repository di Docker.

Passaggio 2: scegli il metodo per installare la patch

Esistono tre metodi per applicare le patch e installare gli aggiornamenti di sicurezza sui nodi. Il metodo che puoi utilizzare dipende dal fatto che la patch sia disponibile o meno nel sistema operativo nel gestore di pacchetti:

1. Per installare le patch con `nodeadm upgrade` pubblicate nei gestori di pacchetti, consulta Step 2 a.
2. Per installare direttamente le patch con i gestori di pacchetti, consulta Step 2 b.
3. Per installare patch personalizzate che non sono pubblicate nei gestori di pacchetti. Nota che ci sono considerazioni speciali per le patch personalizzate per `containerd`, Step 2 c.

### Fase 2 a: Applicazione di patch con l'aggiornamento nodeadm

Dopo aver confermato che la patch CVE `containerd` è stata pubblicata nel sistema operativo o nei repository di Docker (Apt o RPM), puoi utilizzare il comando `nodeadm upgrade` per eseguire

l'aggiornamento alla versione più recente di `containerd`. Poiché non si tratta di un aggiornamento della versione di Kubernetes, devi passare alla versione corrente di Kubernetes al comando `nodeadm` upgrade.

```
nodeadm upgrade K8S_VERSION --config-source file:///root/nodeConfig.yaml
```

Passaggio 2 b: applicazione di patch con i gestori di pacchetti del sistema operativo

In alternativa puoi eseguire l'aggiornamento anche tramite il rispettivo gestore di pacchetti e utilizzarlo per aggiornare il pacchetto `containerd` come segue.

Amazon Linux 2023

```
sudo yum update -y
sudo yum install -y containerd
```

RHEL

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-
ce.repo
sudo yum update -y
sudo yum install -y containerd
```

Ubuntu

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/
docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
 https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update -y
sudo apt install -y --only-upgrade containerd.io
```

Fase 2 c: Patch CVE **`Containerd`** non pubblicata nei gestori di pacchetti

Se la `containerd` versione con patch è disponibile solo con altri mezzi anziché nel gestore di pacchetti, ad esempio nelle GitHub versioni, è possibile installarla `containerd` dal sito ufficiale. GitHub

1. Se il computer è già entrato a far parte del cluster come nodo ibrido, devi eseguire il comando `nodeadm uninstall`.
2. Installa i binari ufficiali `containerd`. È possibile utilizzare i passaggi di installazione ufficiali. GitHub
3. Esegui il comando `nodeadm install` con l'argomento `--containerd-source` impostato su `none`, che salterà l'installazione di `containerd` fino a `nodeadm`. Puoi utilizzare il valore di `none` nella sorgente `containerd` per qualsiasi sistema operativo in esecuzione sul nodo.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --containerd-source
 none
```

## Rimuovi i nodi ibridi

In questa sezione viene descritto come eliminare i nodi ibridi dal cluster Amazon EKS. Devi eliminare i nodi ibridi utilizzando strumenti compatibili con Kubernetes di propria scelta, come kubectl. I costi per i nodi ibridi si interrompono quando l'oggetto nodo viene rimosso dal cluster Amazon EKS. Per ulteriori informazioni sui prezzi dei nodi ibridi, consulta i Prezzi di Amazon EKS.

> **Important**
>
> La rimozione dei nodi interrompe i carichi di lavoro in esecuzione sul nodo. Prima di eliminare i nodi ibridi, ti consigliamo innanzitutto di svuotare il nodo per spostare i pod su un altro nodo attivo. Per ulteriori informazioni su come svuotare i nodi, consulta Safely Drain a Node nella documentazione Kubernetes.

Esegui i seguenti passaggi di kubectl dal computer o dall'istanza locale che usi per interagire con l'endpoint API Kubernetes del cluster Amazon EKS. Se stai usando un file `kubeconfig` specifico, usa il flag `--kubeconfig`.

Passaggio 1: elenca i nodi

```
kubectl get nodes
```

Passaggio 2: svuota il nodo

Consulta kubectl drain nella documentazione di Kubernetes per maggiori informazioni sul comando `kubectl drain`.

```
kubectl drain --ignore-daemonsets <node-name>
```

Passaggio 3: arrestare e disinstallare gli artefatti dei nodi ibridi

Puoi utilizzare il CLI Amazon EKS Hybrid Nodes (`nodeadm`) per arrestare e rimuovere gli artefatti dei nodi ibridi dall'host. Devi eseguire `nodeadm` con un utente che abbia privilegi root/sudo. Per impostazione predefinita, `nodeadm uninstall` non proseguirà se sul nodo rimangono dei pod. Se utilizzi AWS Systems Manager (SSM) come provider di credenziali, il comando `nodeadm uninstall` annulla la registrazione dell'host come istanza gestita AWS SSM. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
nodeadm uninstall
```

Passaggio 4: eliminare il nodo dal cluster

Una volta interrotti e disinstallati gli artefatti dei nodi ibridi, rimuovi la risorsa del nodo dal cluster.

```
kubectl delete node <node-name>
```

Passaggio 5: verifica la presenza di artefatti rimanenti

A seconda del CNI scelto, potrebbero rimanere degli artefatti sui nodi ibridi dopo aver eseguito i passaggi precedenti. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

## Configurazione di reti di applicazioni, componenti aggiuntivi e webhook per nodi ibridi

Dopo aver creato un cluster EKS per nodi ibridi, configura funzionalità aggiuntive per le reti di applicazioni (CNI, BGP, ingresso, bilanciamento del carico, policy di rete), i componenti aggiuntivi, i webhook e le impostazioni proxy. Per l'elenco completo dei componenti aggiuntivi EKS e della

community compatibili con i nodi ibridi, consulta the section called "Configurare i componenti aggiuntivi".

Approfondimenti del cluster EKS EKS include il rilevamento approfondito delle configurazioni errate nella configurazione dei nodi ibridi che potrebbero compromettere la funzionalità del cluster o dei carichi di lavoro. Per ulteriori informazioni sugli approfondimenti del cluster, consulta the section called "Approfondimenti sui cluster".

Di seguito sono elencate le funzionalità e i componenti aggiuntivi comuni puoi utilizzare con i nodi ibridi:

- Container Networking Interface (CNI): AWS supporta Cilium come la CNI per i nodi ibridi. Per ulteriori informazioni, consulta the section called "Configurazione della CNI". Tieni presente che la CNI di AWS VPC non può essere utilizzata con nodi ibridi.
- CoredNS e **`kube-proxy`**: CoreDNS e `kube-proxy` vengono installati automaticamente quando i nodi ibridi si uniscono al cluster EKS. Questi componenti aggiuntivi possono essere gestiti come componenti aggiuntivi EKS dopo la creazione del cluster.
- Ingresso e bilanciamento del carico: puoi utilizzare AWS Load Balancer Controller e Application Load Balancer (ALB) o Network Load Balancer (NLB) con il tipo di destinazione `ip` per i carichi di lavoro in esecuzione su nodi ibridi. AWS supporta le funzionalità integrate di bilanciamento del carico ingresso, gateway e del servizio Kubernetes di Cilium per i carichi di lavoro in esecuzione su nodi ibridi. Per ulteriori informazioni, consultare the section called "Configurazione dell'ingresso" e the section called "Configurazione dei servizi LoadBalancer".
- Metriche: puoi utilizzare gli scraper senza agenti del servizio gestito da Amazon per Prometheus (AMP), AWS Distro per Open Telemetry (ADOT) e l'agente di Amazon CloudWatch Observability con nodi ibridi. Per utilizzare gli scraper senza agenti AMP per le metriche dei pod sui nodi ibridi, i pod devono essere accessibili dal VPC utilizzato per il cluster EKS.
- Log: puoi abilitare la registrazione del piano di controllo (control-plane) EKS per i cluster abilitati ai nodi ibridi. Puoi utilizzare i componenti aggiuntivi ADOT e l'agente di Amazon CloudWatch Observability di EKS per la registrazione ibrida di nodi e pod.
- Pod Identity e IRSA: puoi utilizzare Pod Identity EKS e ruoli IAM per gli account del servizio (IRSA) con applicazioni in esecuzione su nodi ibridi per consentire l'accesso granulare ai pod in esecuzione su nodi ibridi con altri servizi AWS.
- Webhook: se utilizzi webhook, consulta the section called "Configurazione di webhook" per le considerazioni e i passaggi per eseguire i webhook sui nodi cloud, se non riesci a rendere instradabili le reti di pod on-premises.

- Proxy: se utilizzi un server proxy nell'ambiente on-premises per il traffico in uscita dal data center o dall'ambiente edge, puoi configurare i nodi e il cluster ibridi per usare il server proxy. Per ulteriori informazioni, consulta the section called "Configurazione del proxy".

Argomenti


- Configurazione della CNI per nodi ibridi
- Configurare componenti aggiuntivi per nodi ibridi
- Configurazione di webhook per nodi ibridi
- Configurare il proxy per i nodi ibridi
- Configurazione di Cilium BGP per nodi ibridi
- Configurazione dell'ingresso Kubernetes per nodi ibridi
- Configurazione dei servizi di tipo LoadBalancer per nodi ibridi
- Configurazione delle policy di rete Kubernetes per i nodi ibridi


## Configurazione della CNI per nodi ibridi

Cilium è la Container Networking Interface (CNI) AWS supportata per Amazon EKS Hybrid Nodes. Devi installare una CNI affinché i nodi ibridi siano pronti a servire i carichi di lavoro. I nodi ibridi vengono visualizzati con lo stato `Not Ready` fino all'esecuzione di una CNI. Puoi gestire la CNI con strumenti a tua scelta come Helm. Le istruzioni in questa pagina riguardano la gestione del ciclo di vita di Cilium (installazione, aggiornamento, eliminazione). Consulta the section called "Panoramica dell'ingresso di Cilium e del gateway di Cilium", the section called "Tipo di servizio LoadBalancer" e the section called "Configurazione delle policy di rete" per scoprire come configurare Cilium per l'ingresso, il bilanciamento del carico e le policy di rete.

Cilium non è supportato da AWS quando viene eseguito su nodi in Cloud. AWS La CNI di Amazon VPC non è compatibile con i nodi ibridi e la CNI del VPC è configurata con la non-affinità per l'etichetta `eks.amazonaws.com/compute-type: hybrid`.

La documentazione di Calico precedentemente contenuta in questa pagina è stata spostata in EKS Hybrid Examples Repository.

Compatibilità delle versioni

Le versioni `v1.17.x` di Cilium `v1.18.x` sono supportate per EKS Hybrid Nodes per ogni versione di Kubernetes supportata in Amazon EKS.

> **Note**
>
> Requisiti del kernel Cilium v1.18.3: a causa dei requisiti del kernel (kernel Linux >= 5.10), Cilium v1.18.3 non è supportato su:

- Ubuntu 20.04
- Red Hat Enterprise Linux (RHEL) 8

Per i requisiti di sistema, consulta i requisiti di sistema di Cilium.

Consulta il supporto delle versioni Kubernetes per individuare quali sono le versioni supportate da Amazon EKS. EKS Hybrid Nodes ha lo stesso supporto per le versioni di Kubernetes dei cluster Amazon EKS con nodi cloud.

Funzionalità supportate

AWS mantiene versioni di Cilium for EKS Hybrid Nodes basate sul progetto open source Cilium. Per ricevere supporto da AWS Cilium, è necessario utilizzare le build Cilium AWS mantenute e le versioni di Cilium supportate.

AWS fornisce supporto tecnico per le configurazioni predefinite delle seguenti funzionalità di Cilium da utilizzare con EKS Hybrid Nodes. Se si prevede di utilizzare funzionalità che non rientrano nell'ambito del AWS supporto, si consiglia di ottenere un supporto commerciale alternativo per Cilium o di disporre dell'esperienza interna necessaria per risolvere i problemi e contribuire alle correzioni del progetto Cilium.

| Funzionalità di Cilium | Supportato da AWS |
|---|---|
| Conformità della rete Kubernetes | Sì |
| Connettività del cluster principale | Sì |
| Famiglia di IP | IPv4 |
| Gestione del ciclo di vita | Helm |
| Modalità di rete | Incapsulamento VXLAN |

| Funzionalità di Cilium | Supportato da AWS |
|---|---|
| Gestione indirizzi IP (IPAM) | Ambito del cluster IPAM (Gestione degli indirizzi IP) di Cilium |
| Policy di rete | Policy di rete Kubernetes |
| Border Gateway Protocol (BGP) | Piano di controllo (control-plane) BGP di Cilium |
| Ingresso Kubernetes | Ingresso di Cilium, gateway di Cilium |
| Allocazione LoadBalancer IP del servizio | IPAM del bilanciatore del carico di Cilium |
| Pubblicità LoadBalancer dell'indirizzo IP del servizio | Piano di controllo (control-plane) BGP di Cilium |
| sostituzione kube-proxy | Sì |

Considerazioni su Cilium

- Repository Helm: AWS ospita il grafico Cilium Helm nell'Amazon Elastic Container Registry Public (Amazon ECR Public) presso Amazon EKS Cilium/Cilium. Le versioni disponibili includono:
  - Cilium v1.17.9: `oci://public.ecr.aws/eks/cilium/cilium:1.17.9-0`
  - Cilium v1.18.3: `oci://public.ecr.aws/eks/cilium/cilium:1.18.3-0`

    I comandi in questo argomento utilizzano questo repository. Tieni presente che alcuni comandi `helm repo` non sono validi per i repository Helm in Amazon ECR Public, quindi non puoi fare riferimento a questo repository da un nome di repository Helm locale. Invece, utilizza l'URI completo nella maggior parte dei comandi.
- Per impostazione predefinita, Cilium è configurato per l'esecuzione in modalità sovrapposizione/tunnel con VXLAN come metodo di incapsulamento. Questa modalità ha il minor numero di requisiti sulla rete fisica sottostante.
- Per impostazione predefinita, Cilium maschera l'indirizzo IP di origine di tutto il traffico dei pod in uscita dal cluster all'indirizzo IP del nodo. Se disabiliti il mascheramento, il tuo pod CIDRs deve essere instradabile sulla tua rete locale.

- Se esegui webhook su nodi ibridi, il pod CIDRs deve essere instradabile sulla rete locale. Se i pod non CIDRs sono instradabili sulla rete locale, si consiglia di eseguire webhook sui nodi cloud dello stesso cluster. Per ulteriori informazioni, consulta the section called "Configurazione di webhook" e the section called "Preparazione della rete".
- AWS consiglia di utilizzare la funzionalità BGP integrata di Cilium per rendere il pod instradabile sulla rete locale. CIDRs Per ulteriori informazioni su come configurare il BGP di Cilium con nodi ibridi, consulta the section called "Configurazione di BGP".
- La gestione degli indirizzi IP (IPAM) predefinita in Cilium si chiama Cluster Scope, in cui l'operatore Cilium alloca gli indirizzi IP per ogni nodo in base al pod configurato dall'utente. CIDRs

Installazione di Cilium su nodi ibridi

Procedura

1. Crea un file YAML denominato `cilium-values.yaml`. L'esempio seguente configura Cilium per l'esecuzione solo su nodi ibridi impostando l'affinità per l'etichetta `eks.amazonaws.com/compute-type: hybrid` per l'agente e l'operatore di Cilium.
   - Configura *`clusterPoolIpv4PodCIDRList`* con lo stesso pod che CIDRs hai configurato per le reti di pod remoti del tuo cluster EKS. Ad esempio, `10.100.0.0/24`. L'operatore di Cilium alloca le porzioni di indirizzi IP dall'interno dello spazio IP `clusterPoolIpv4PodCIDRList` configurato. Il CIDR del pod non deve sovrapporsi al nodo CIDR on-premises, al CIDR del VPC o del servizio Kubernetes.
   - Configura `clusterPoolIpv4MaskSize` in base ai pod richiesti per nodo. Ad esempio, `25` per una dimensione dei segmenti di /25 di 128 pod per nodo.
   - Non modificare `clusterPoolIpv4PodCIDRList` o `clusterPoolIpv4MaskSize` dopo aver implementato Cilium sul cluster, consulta Expanding the cluster pool per ulteriori informazioni.
   - Se esegui Cilium in modalità sostitutiva kube-proxy, imposta `kubeProxyReplacement: "true"` nei valori Helm e assicurati di non avere un'implementazione kube-proxy esistente in esecuzione sugli stessi nodi di Cilium.
   - L'esempio seguente disabilita il proxy Envoy Layer 7 (L7) utilizzato da Cilium per le policy di rete e l'ingresso L7. Per ulteriori informazioni, consultare the section called "Configurazione delle policy di rete" e the section called "Panoramica dell'ingresso di Cilium e del gateway di Cilium".
   - L'esempio seguente configura `loadBalancer.serviceTopology: true` affinché la distribuzione del traffico del servizio funzioni correttamente se configurata per i tuoi servizi. Per

ulteriori informazioni, consulta the section called "Configurazione della distribuzione del traffico del servizio".

- Per un elenco completo dei valori Helm per Cilium, consulta Helm reference nella documentazione di Cilium.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: In
          values:
          - hybrid
ipam:
  mode: cluster-pool
  operator:
    clusterPoolIPv4MaskSize: 25
    clusterPoolIPv4PodCIDRList:
    - POD_CIDR
loadBalancer:
  serviceTopology: true
operator:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
              - hybrid
  unmanagedPodWatcher:
    restart: false
loadBalancer:
  serviceTopology: true
envoy:
  enabled: false
kubeProxyReplacement: "false"
```

2. Installa Cilium nel cluster.

- Sostituisci `CILIUM_VERSION` con una versione Cilium (ad esempio `1.17.9-0` o `1.18.3-0`). Consigliamo di utilizzare la versione della patch più recente per la versione secondaria di Cilium.
- Assicurati che i tuoi nodi soddisfino i requisiti del kernel per la versione che scegli. Cilium v1.18.3 richiede un kernel Linux >= 5.10.
- Se stai utilizzando un file kubeconfig specifico, usa il flag `--kubeconfig` con il comando di installazione Helm.

```
helm install cilium oci://public.ecr.aws/eks/cilium/cilium \
    --version CILIUM_VERSION \
    --namespace kube-system \
    --values cilium-values.yaml
```

3. Conferma che l'installazione di Cilium sia stata eseguita correttamente con i comandi seguenti. Dovresti visualizzare l'implementazione `cilium-operator` e `cilium-agent` in esecuzione su ciascuno dei tuoi nodi ibridi. Inoltre, i nodi ibridi dovrebbero avere lo stato `Ready`. Per informazioni su come configurare Cilium BGP per pubblicizzare il pod sulla rete locale, procedi a. CIDRs the section called "Configurazione di BGP"

```
kubectl get pods -n kube-system
```

```
NAME                              READY   STATUS    RESTARTS   AGE
cilium-jjjn8                      1/1     Running   0          11m
cilium-operator-d4f4d7fcb-sc5xn   1/1     Running   0          11m
```

```
kubectl get nodes
```

```
NAME                   STATUS   ROLES    AGE   VERSION
mi-04a2cf999b7112233   Ready    <none>   19m   v1.31.0-eks-a737599
```

Aggiornamento di Cilium su nodi ibridi

Prima di aggiornare l'implementazione di Cilium, consulta attentamente la documentazione sull'aggiornamento di Cilium e le note sull'aggiornamento per comprendere le modifiche nella versione di Cilium di destinazione.

1. Assicurati di aver installato la CLI `helm` nel tuo ambiente a riga di comando. Per istruzioni sull'installazione, consulta la documentazione di Helm.

2. Esegui il controllo preflight dell'aggiornamento di Cilium. Sostituisci `CILIUM_VERSION` con la versione di Cilium di destinazione. Ti consigliamo di eseguire la versione della patch più recente per la versione secondaria di Cilium. Puoi trovare la versione della patch più recente per una determinata versione minore di Cilium nella sezione Stable Releases della documentazione di Cilium.

```
helm install cilium-preflight oci://public.ecr.aws/eks/cilium/cilium --version
 CILIUM_VERSION \
  --namespace=kube-system \
  --set preflight.enabled=true \
  --set agent=false \
  --set operator.enabled=false
```

3. Dopo aver applicato `cilium-preflight.yaml`, assicurati che il numero di pod `READY` sia lo stesso numero di pod di Cilium in esecuzione.

```
kubectl get ds -n kube-system | sed -n '1p;/cilium/p'
```

```
NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
cilium                    2         2         2       2            2           <none>
          1h20m
cilium-pre-flight-check   2         2         2       2            2           <none>
          7m15s
```

4. Una volta che il numero di pod READY è uguale, assicurati che anche l'implementazione preflight di Cilium sia contrassegnata come READY 1/1. Se viene visualizzato READY 0/1, consulta la sezione di CNP Validation e risolvi i problemi relativi all'implementazione prima di continuare con l'aggiornamento.

```
kubectl get deployment -n kube-system cilium-pre-flight-check -w
```

```
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
cilium-pre-flight-check   1/1     1            0           12s
```

5. Elimina il preflight

```
helm uninstall cilium-preflight --namespace kube-system
```

6. Prima di eseguire il comando `helm upgrade`, conserva i valori dell'implementazione in `existing-cilium-values.yaml` o utilizza le opzioni `--set` della riga di comando per le impostazioni quando esegui il comando di aggiornamento. L'operazione di aggiornamento sovrascrive Cilium ConfigMap, quindi è fondamentale che i valori di configurazione vengano trasmessi durante l'aggiornamento.

```
helm get values cilium --namespace kube-system -o yaml > existing-cilium-values.yaml
```

7. Durante le normali operazioni del cluster, tutti i componenti Cilium devono eseguire la stessa versione. I passaggi seguenti descrivono come aggiornare tutti i componenti da una versione stabile a una successiva. Quando esegui l'aggiornamento da una versione secondaria a un'altra, consigliamo di aggiornare prima alla versione della patch più recente per la versione secondaria esistente di Cilium. Per ridurre al minimo le interruzioni, imposta l'opzione `upgradeCompatibility` sulla versione iniziale di Cilium installata in questo cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium --version CILIUM_VERSION \
  --namespace kube-system \
  --set upgradeCompatibility=1.X \
  -f existing-cilium-values.yaml
```

8. (Facoltativo) Se devi eseguire il rollback dell'aggiornamento a causa di problemi, esegui i comandi seguenti.

```
helm history cilium --namespace kube-system
helm rollback cilium [REVISION] --namespace kube-system
```

Eliminazione di Cilium dai nodi ibridi

1. Esegui il comando seguente per disinstallare tutti i componenti Cilium dal tuo cluster. Nota, la disinstallazione della CNI potrebbe influire sulla salute di nodi e pod e non dovrebbe essere eseguita sui cluster di produzione.

```
helm uninstall cilium --namespace kube-system
```

Le interfacce e le rotte configurate da Cilium non vengono rimosse per impostazione predefinita quando il CNI viene rimosso dal cluster, consulta il GitHub problema per ulteriori informazioni.

2. Per ripulire i file e le risorse di configurazione su disco, se si utilizzano le directory di configurazione standard, è possibile rimuovere i file come mostrato dallo `cni-uninstall.sh`script nel repository Cilium su. GitHub
3. Per rimuovere Cilium Custom Resource Definitions (CRDs) dal tuo cluster, puoi eseguire i seguenti comandi.

```
kubectl get crds -oname | grep "cilium" | xargs kubectl delete
```

## Configurare componenti aggiuntivi per nodi ibridi

Questa pagina descrive le considerazioni sull'esecuzione di componenti aggiuntivi e AWS componenti aggiuntivi della community su Amazon EKS Hybrid Nodes. Per ulteriori informazioni sui componenti aggiuntivi di Amazon EKS e sui processi per la creazione, l'aggiornamento e la rimozione di componenti aggiuntivi dal cluster, consulta the section called "Componenti aggiuntivi Amazon EKS". Salvo diversa indicazione in questa pagina, i processi per la creazione, l'aggiornamento e la rimozione dei componenti aggiuntivi Amazon EKS sono gli stessi per i cluster Amazon EKS con nodi ibridi e per i cluster Amazon EKS con nodi in esecuzione nel cloud. AWS Solo i componenti aggiuntivi inclusi in questa pagina sono stati convalidati per la compatibilità con Amazon EKS Hybrid Nodes.

I seguenti AWS componenti aggiuntivi sono compatibili con Amazon EKS Hybrid Nodes.

| AWS componente aggiuntivo | Versioni dei componenti aggiuntivi compatibili |
|---|---|
| kube-proxy | v1.25.14-eksbuild.2 e versioni successive |
| CoreDNS | v1.9.3-eksbuild.7 e versioni successive |
| AWS Distro per OpenTelemetry (ADOT) | v0.102.1-eksbuild.2 e versioni successive |
| CloudWatch Agente di osservabilità | v2.2.1-eksbuild.1 e versioni successive |
| EKS Pod Identity Agent | • v1.3.3-eksbuild.1 e versioni successive, ad eccezione di Bottlerocket<br>• v1.3.7-eksbuild.2 e versioni successive per Bottlerocket |
| Agente di monitoraggio del nodo | v1.2.0-eksbuild.1 e versioni successive |

| AWS componente aggiuntivo | Versioni dei componenti aggiuntivi compatibili |
|---|---|
| Controller di snapshot CSI | v8.1.0-eksbuild.1 e versioni successive |
| AWS Connettore CA privato per Kubernetes | v1.6.0-eksbuild.1 e versioni successive |
| Driver Amazon FSx CSI | v1.7.0-eksbuild.1 e versioni successive |
| AWS Fornitore di driver CSI Secrets Store | v2.1.1-eksbuild.1 e versioni successive |

I seguenti componenti aggiuntivi della community sono compatibili con Amazon EKS Hybrid Nodes. Per ulteriori informazioni sui componenti aggiuntivi della community, consultathe section called "Componenti aggiuntivi della community".

| Componente aggiuntivo | Versioni dei componenti aggiuntivi compatibili |
|---|---|
| Kubernetes Metrics Server | v0.7.2-eksbuild.1 e versioni successive |
| cert-manager | v1.17.2-eksbuild.1 e versioni successive |
| Prometheus Node Exporter | v1.9.1-eksbuild.2 e versioni successive |
| kube-state-metrics | v2.15.0-eksbuild.4 e versioni successive |
| DNS esterno | v0.19.0-eksbuild.1 e versioni successive |

Oltre ai componenti aggiuntivi Amazon EKS indicati nelle tabelle precedenti, il Collettore Amazon Managed Service for Prometheus e il AWS Load Balancer Controller per l'ingresso delle applicazioni (HTTP) e il bilanciamento del carico (TCP/UDP) sono compatibili con i nodi ibridi.

Esistono AWS componenti aggiuntivi e componenti aggiuntivi della community che non sono compatibili con Amazon EKS Hybrid Nodes. Le versioni più recenti di questi componenti aggiuntivi prevedono una regola di antiaffinità per l'etichetta predefinita `eks.amazonaws.com/compute-type: hybrid` applicata ai nodi ibridi. Ciò impedisce loro di funzionare su nodi ibridi quando vengono distribuiti nei cluster. Se disponi di cluster con nodi ibridi e nodi in esecuzione nel AWS cloud, puoi distribuire questi componenti aggiuntivi nel cluster su nodi in esecuzione nel cloud. AWS Amazon VPC CNI non è compatibile con i nodi ibridi e Cilium e Calico sono supportati come

Container Networking Interfaces (CNIs) per i nodi ibridi Amazon EKS. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

AWS componenti aggiuntivi

Le sezioni seguenti descrivono le differenze tra l'esecuzione di AWS componenti aggiuntivi compatibili su nodi ibridi rispetto ad altri tipi di elaborazione Amazon EKS.

kube-proxy e CoreDNS

EKS installa kube-proxy e CoredNS come componenti aggiuntivi autogestiti per impostazione predefinita quando crei un cluster EKS con l'API e, anche dalla CLI. AWS AWS SDKs AWS É possibile sovrascrivere questi componenti aggiuntivi con i componenti aggiuntivi Amazon EKS dopo la creazione del cluster. Fai riferimento alla documentazione EKS per i dettagli su the section called "kube-proxy" e the section called "CoreDNS". Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi nel AWS cloud, AWS consiglia di avere almeno una replica CoreDNS sui nodi ibridi e almeno una replica CoredNS sui nodi nel cloud. AWS Per i passaggi di configurazione, consulta la sezione the section called "Configurazione delle repliche CoreDNS".

CloudWatch Agente di osservabilità

L'operatore dell'agente CloudWatch Observability utilizza i webhook. Se esegui l'operatore su nodi ibridi, il tuo pod CIDR on-premises deve essere instradabile sulla rete on-premises e devi configurare il cluster EKS con la tua rete di pod remota. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

Le metriche a livello di nodo non sono disponibili per i nodi ibridi perché CloudWatch Container Insights dipende dalla disponibilità di Instance Metadata Service (IMDS) per le metriche a livello di nodo. Per i nodi ibridi sono disponibili metriche a livello di cluster, carico di lavoro, pod e container.

Dopo aver installato il componente aggiuntivo seguendo i passaggi descritti in Installa l' CloudWatch agente con Amazon CloudWatch Observability, il manifesto del componente aggiuntivo deve essere aggiornato prima che l'agente possa essere eseguito correttamente sui nodi ibridi. Modifica la risorsa `amazoncloudwatchagents` sul cluster per aggiungere la variabile di ambiente `RUN_WITH_IRSA` come mostrato di seguito.

```
kubectl edit amazoncloudwatchagents -n amazon-cloudwatch cloudwatch-agent
```

```
apiVersion: v1
```

```
items:
- apiVersion: cloudwatch.aws.amazon.com/v1alpha1
  kind: AmazonCloudWatchAgent
  metadata:
    ...
    name: cloudwatch-agent
    namespace: amazon-cloudwatch
    ...
  spec:
    ...
    env:
    - name: RUN_WITH_IRSA # <-- Add this
      value: "True" # <-- Add this
    - name: K8S_NODE_NAME
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
          ...
```

Raccoglitore gestito di Amazon Managed Prometheus per nodi ibridi

Un servizio gestito da Amazon per Prometheus (AMP) è costituito da uno scraper che rileva e raccoglie i parametri da un cluster Amazon EKS. AMP gestisce lo scraper per te, eliminando la necessità di gestire personalmente istanze, agenti o scraper.

Puoi utilizzare i raccoglitori gestiti AMP senza alcuna configurazione aggiuntiva specifica per i nodi ibridi. Tuttavia, gli endpoint metrici per le applicazioni sui nodi ibridi devono essere raggiungibili dal VPC, compresi i percorsi dal VPC alla rete pod remota CIDRs e le porte aperte nel firewall locale. Inoltre, il cluster deve disporre di un accesso privato agli endpoint del cluster.

Segui i passaggi descritti in Utilizzo di un raccoglitore AWS gestito nella Guida per l'utente di Amazon Managed Service for Prometheus.

AWS Distro per (ADOT) OpenTelemetry

Puoi utilizzare il componente aggiuntivo AWS Distro for OpenTelemetry (ADOT) per raccogliere metriche, log e dati di tracciamento dalle tue applicazioni in esecuzione su nodi ibridi. ADOT utilizza i webhook di ammissione per mutare e convalidare le richieste delle risorse personalizzate del raccoglitore. Se esegui l'operatore ADOT su nodi ibridi, il tuo pod CIDR on-premises deve essere instradabile sulla rete on-premises e devi configurare il cluster EKS con la tua rete di pod remota. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

Segui la procedura descritta in Guida introduttiva a AWS Distro per OpenTelemetry utilizzare i componenti aggiuntivi EKS nella Distro per la documentazione. AWS OpenTelemetry

AWS Controller Load Balancer

È possibile utilizzare AWS Load Balancer Controller e Application Load Balancer (ALB) o Network Load Balancer (NLB) con il `ip` tipo di destinazione per i carichi di lavoro sui nodi ibridi. Le destinazioni IP utilizzate con ALB o NLB devono essere instradabili da. AWSIl controller AWS Load Balancer utilizza anche i webhook. Se esegui l'operatore AWS Load Balancer Controller su nodi ibridi, il tuo pod CIDR locale deve essere instradabile sulla tua rete locale e devi configurare il tuo cluster EKS con la tua rete di pod remoti. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

Per installare il AWS Load Balancer Controller, segui i passaggi indicati in the section called "AWS Application Load Balancer" o. the section called "Network Load Balancer di AWS"

Per l'accesso con ALB, è necessario specificare le annotazioni riportate di seguito. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di applicazione".

```
alb.ingress.kubernetes.io/target-type: ip
```

Per il bilanciamento del carico con NLB, è necessario specificare le annotazioni riportate di seguito. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di rete".

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

EKS Pod Identity Agent

> **Note**
>
> Per implementare correttamente il componente aggiuntivo EKS Pod Identity Agent su nodi ibridi che eseguono Bottlerocket, assicurati che la versione di Bottlerocket sia almeno la v1.39.0. Il Pod Identity Agent non è supportato nelle versioni precedenti di Bottlerocket in ambienti con nodi ibridi.

L'originale Amazon EKS Pod Identity Agent DaemonSet si basa sulla disponibilità di EC2 IMDS sul nodo per ottenere le credenziali richieste. AWS Poiché IMDS non è disponibile sui nodi ibridi, a partire dalla versione 1.3.3-eksbuild.1, il componente aggiuntivo Pod Identity Agent implementa

facoltativamente un componente che monta le credenziali richieste. DaemonSet I nodi ibridi che eseguono Bottlerocket richiedono un metodo diverso per montare le credenziali e, a partire dalla versione 1.3.7-eksbuild.2, il componente aggiuntivo Pod Identity Agent ne implementa facoltativamente uno che si rivolge specificamente ai nodi ibridi Bottlerocket. DaemonSet Le sezioni DaemonSets seguenti descrivono il processo per abilitare l'opzione.

Ubuntu/RHEL/AL2023

1. Per utilizzare l'agente Pod Identity sui nodi ibridi Ubuntu/RHEL/Al 2023, `enableCredentialsFile: true` impostalo nella sezione ibrida di `nodeadm` config come mostrato di seguito:

   ```
   apiVersion: node.eks.aws/v1alpha1
   kind: NodeConfig
   spec:
       hybrid:
           enableCredentialsFile: true # <-- Add this
   ```

   Questo configurerà `nodeadm` per creare un file di credenziali da configurare sul nodo sotto `/eks-hybrid/.aws/credentials`, che verrà utilizzato dai pod `eks-pod-identity-agent`. Questo file di credenziali conterrà AWS credenziali temporanee che verranno aggiornate periodicamente.

2. Dopo aver aggiornato la configurazione `nodeadm` su ogni nodo, esegui il seguente comando `nodeadm init` con il tuo `nodeConfig.yaml` per unire i nodi ibridi al cluster Amazon EKS. Se i tuoi nodi si sono uniti al cluster in precedenza, esegui comunque di nuovo il comando `nodeadm init`.

   ```
   nodeadm init -c file://nodeConfig.yaml
   ```

3. Installa `eks-pod-identity-agent` con il supporto per i nodi ibridi abilitato, utilizzando la AWS CLI o. Console di gestione AWS

   a. AWS CLI: dalla macchina che stai utilizzando per amministrare il cluster, esegui il seguente comando per l'installazione `eks-pod-identity-agent` con il supporto per i nodi ibridi abilitato. Sostituisci `my-cluster` con il nome del cluster.

      ```
      aws eks create-addon \
          --cluster-name my-cluster \
          --addon-name eks-pod-identity-agent \
          --configuration-values '{"daemonsets":{"hybrid":{"create": true}}}'
      ```

b. Console di gestione AWS: Se stai installando il componente aggiuntivo Pod Identity Agent tramite la AWS console, aggiungi quanto segue alla configurazione opzionale per distribuire i DaemonSet nodi ibridi destinati.

```
{"daemonsets":{"hybrid":{"create": true}}}
```

Bottlerocket

1. Per utilizzare l'agente Pod Identity sui nodi ibridi Bottlerocket, aggiungi il flag `--enable-credentials-file=true` al comando utilizzato per i dati utente del contenitore bootstrap Bottlerocket, come descritto in the section called "Connessione dei nodi ibridi con Bottlerocket".

   a. Se stai usando il provider di credenziali SSM, il tuo comando dovrebbe avere il seguente aspetto:

   ```
   eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-
   code=<activation-code> --region=<region> --enable-credentials-file=true
   ```

   b. Se stai usando il provider di credenziali IAM Roles Anywhere, il tuo comando dovrebbe avere il seguente aspetto:

   ```
   eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key> --enable-
   credentials-file=true
   ```

   Questo configurerà lo script di bootstrap per creare un file di credenziali sul nodo sotto `/var/eks-hybrid/.aws/credentials`, che verrà utilizzato dai pod `eks-pod-identity-agent`. Questo file di credenziali conterrà AWS credenziali temporanee che verranno aggiornate periodicamente.

2. Installa `eks-pod-identity-agent` con il supporto per i nodi ibridi Bottlerocket abilitato, utilizzando la CLI AWS o. Console di gestione AWS

   a. AWS CLI: dalla macchina che stai utilizzando per amministrare il cluster, esegui il seguente comando per l'installazione `eks-pod-identity-agent` con il supporto per i nodi ibridi Bottlerocket abilitato. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks create-addon \
       --cluster-name my-cluster \
       --addon-name eks-pod-identity-agent \
   ```

```
    --configuration-values '{"daemonsets":{"hybrid-bottlerocket":{"create":
true}}}'
```

b. Console di gestione AWS: Se stai installando il componente aggiuntivo Pod Identity Agent tramite la AWS console, aggiungi quanto segue alla configurazione opzionale per distribuire i nodi ibridi Bottlerocket destinati. DaemonSet

```
{"daemonsets":{"hybrid-bottlerocket":{"create": true}}}
```

Controller di snapshot CSI

A partire dalla versione`v8.1.0-eksbuild.2`, il componente aggiuntivo del controller snapshot CSI applica una regola di antiaffinità morbida per i nodi ibridi, preferendo che il controller `deployment` venga eseguito su EC2 nella stessa regione del piano di controllo di Amazon AWS EKS. La collocazione congiunta `deployment` nella stessa AWS regione del piano di controllo di Amazon EKS migliora la latenza.

Componenti aggiuntivi della community

Le sezioni seguenti descrivono le differenze tra l'esecuzione di componenti aggiuntivi della community compatibili su nodi ibridi rispetto ad altri tipi di elaborazione Amazon EKS.

Kubernetes Metrics Server

Il piano di controllo deve raggiungere l'IP del pod di Metrics Server (o l'IP del nodo se hostNetwork è abilitato). Pertanto, a meno che non esegui Metrics Server in modalità hostNetwork, devi configurare una rete di pod remota durante la creazione del cluster Amazon EKS e devi rendere instradabili gli indirizzi IP del pod. L'implementazione del Border Gateway Protocol (BGP) con il CNI è un modo comune per rendere instradabili gli indirizzi IP dei pod.

cert-manager

`cert-manager` utilizza i webhook. Se esegui `cert-manager` su nodi ibridi, il tuo pod CIDR on-premises deve essere instradabile sulla rete on-premises e devi configurare il cluster EKS con la tua rete di pod remota. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

## Configurazione di webhook per nodi ibridi

Questa pagina descrive in dettaglio le considerazioni sull'esecuzione di webhook con nodi ibridi. I webhook vengono utilizzati nelle applicazioni Kubernetes e nei progetti open source, come Load

AWS Balancer Controller e CloudWatch Observability Agent, per eseguire funzionalità di mutazione e convalida in fase di esecuzione.

Reti di pod instradabili

Se riesci a rendere il CIDR del tuo pod on-premises instradabile sulla tua rete on-premises, puoi eseguire webhook su nodi ibridi. Esistono diverse tecniche che puoi utilizzare per rendere il CIDR del pod on-premises instradabile sulla tua rete on-premises, tra cui Border Gateway Protocol (BGP), route statiche o altre soluzioni di instradamento personalizzate. BGP è la soluzione consigliata in quanto è più scalabile e più facile da gestire rispetto alle soluzioni alternative che richiedono una configurazione personalizzata o manuale dei percorsi. AWS supporta le funzionalità BGP di Cilium e Calico per la pubblicità dei pod CIDR, vedi e per ulteriori informazioni. the section called "Configurazione della CNI" the section called "CIDR dei pod remoti instradabili"

Reti di pod non instradabili

Se non riesci a rendere il CIDR del tuo pod on-premises instradabile sulla tua rete on-premises e devi eseguire i webhook, ti consigliamo di eseguire tutti i webhook sui nodi cloud nello stesso cluster EKS dei nodi ibridi.

Considerazioni per i cluster in modalità mista

I cluster in modalità mista  sono definiti come cluster EKS con nodi ibridi e nodi in esecuzione nel cloud. AWS Quando esegui un cluster in modalità mista, considera i seguenti consigli:

- Esegui il VPC CNI sui nodi in AWS Cloud e Cilium o Calico sui nodi ibridi. Cilium e Calico non sono supportati da AWS quando vengono eseguiti su nodi in Cloud. AWS
- Configura i webhook per l'esecuzione sui nodi nel cloud. AWS Scopri come configurare the section called "Configurazione dei webhook per i componenti aggiuntivi" i webhook e i componenti aggiuntivi della community. AWS
- Se le tue applicazioni richiedono che i pod in esecuzione su nodi in AWS Cloud comunichino direttamente con i pod in esecuzione su nodi ibridi («comunicazione est-ovest») e utilizzi il VPC CNI sui nodi in AWS Cloud e Cilium o Calico sui nodi ibridi, allora il tuo pod CIDR locale deve essere instradabile sulla tua rete locale.
- Esegui almeno una replica di CoreDNS sui nodi in AWS Cloud e almeno una replica di CoreDNS sui nodi ibridi.
- Configura la distribuzione del traffico del servizio per mantenerlo locale rispetto alla zona da cui proviene. Per ulteriori informazioni sulla Distribuzione del traffico del servizio, consulta the section called "Configurazione della distribuzione del traffico del servizio".

- Se utilizzi AWS Application Load Balancer (ALB) o Network Load Balancer (NLB) per il traffico dei carichi di lavoro in esecuzione su nodi ibridi, allora i target IP utilizzati con ALB o NLB devono essere instradabili da. AWS
- Il componente aggiuntivo Metrics Server richiede la connettività dal piano di controllo EKS all'indirizzo IP del pod Metrics Server. Se esegui il componente aggiuntivo Metrics Server su nodi ibridi, il CIDR del pod on-premises deve essere instradabile sulla rete on-premises.
- Per raccogliere metriche per i nodi ibridi utilizzando il Servizio gestito da Amazon per Prometheus (AMP), il CIDR del pod on-premises deve essere instradabile sulla rete on-premises. In alternativa, puoi utilizzare il collettore gestito AMP per le metriche e le risorse del piano di controllo EKS in esecuzione nel AWS cloud e il componente aggiuntivo AWS Distro for OpenTelemetry (ADOT) per raccogliere le metriche per i nodi ibridi.

Configurazione di cluster in modalità mista

Per visualizzare i webhook in fase di mutazione e convalida in esecuzione sul cluster, puoi visualizzare il tipo di risorsa Estensioni nel pannello Risorse della console EKS per il cluster oppure utilizzare i seguenti comandi. EKS riporta anche le metriche dei webhook nella dashboard di osservabilità del cluster, consulta the section called "Dashboard di osservabilità" per ulteriori informazioni.

```
kubectl get mutatingwebhookconfigurations
```

```
kubectl get validatingwebhookconfigurations
```

Configurazione della distribuzione del traffico del servizio

Quando esegui cluster in modalità mista, ti consigliamo di utilizzare la Distribuzione del traffico del servizio per mantenere il traffico del servizio locale rispetto alla zona da cui proviene. La Distribuzione del traffico del servizio (disponibile per le versioni 1.31 e successive di Kubernetes in EKS) è la soluzione consigliata rispetto al Topology Aware Routing perché è più prevedibile. Con la Distribuzione del traffico del servizio, gli endpoint integri della zona riceveranno tutto il traffico di quella zona. Con il Topology Aware Routing, ciascun servizio deve soddisfare diverse condizioni in quella zona per applicare il routing personalizzato, altrimenti indirizza il traffico in modo uniforme verso tutti gli endpoint.

Se utilizzi Cilium come CNI, devi eseguire la CNI con il set `enable-service-topology` su `true` per abilitare la distribuzione del traffico del servizio. Puoi trasferire questa configurazione con il flag

di installazione Helm `--set loadBalancer.serviceTopology=true` oppure puoi aggiornare un'installazione esistente con il comando CLI `cilium config set enable-service-topology true` di Cilium. L'agente Cilium in esecuzione su ciascun nodo deve essere riavviato dopo aver aggiornato la configurazione per un'installazione esistente.

Nella sezione seguente è riportato un esempio di come configurare la distribuzione del traffico del servizio per il servizio CoreDNS, ti consigliamo di abilitare la stessa opzione per tutti i servizi del cluster per evitare traffico indesiderato tra gli ambienti.

Configurazione delle repliche CoreDNS

Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi nel AWS Cloud, ti consigliamo di avere almeno una replica CoreDNS sui nodi ibridi e almeno una replica CoreDNS sui tuoi nodi nel Cloud. AWS Per evitare problemi di latenza e di rete in una configurazione di cluster in modalità mista, puoi configurare il servizio CoreDNS in modo da preferire la replica CoreDNS più vicina con la Distribuzione del traffico del servizio.

La Distribuzione del traffico del servizio (disponibile per le versioni 1.31 e successive di Kubernetes in EKS) è la soluzione consigliata rispetto al Topology Aware Routing perché è più prevedibile. Nella Distribuzione del traffico del servizio, gli endpoint integri della zona riceveranno tutto il traffico di quella zona. Nel Topology Aware Routing, ciascun servizio deve soddisfare diverse condizioni in quella zona per applicare il routing personalizzato, altrimenti indirizza il traffico in modo uniforme verso tutti gli endpoint. I seguenti passaggi configurano la Distribuzione del traffico del servizio.

Se utilizzi Cilium come CNI, devi eseguire la CNI con il set `enable-service-topology` su `true` per abilitare la Distribuzione del traffico del servizio. Puoi trasferire questa configurazione con il flag di installazione Helm `--set loadBalancer.serviceTopology=true` oppure puoi aggiornare un'installazione esistente con il comando CLI `cilium config set enable-service-topology true` di Cilium. L'agente Cilium in esecuzione su ciascun nodo deve essere riavviato dopo aver aggiornato la configurazione per un'installazione esistente.

1. Aggiungi un'etichetta di zona topologica per ciascuno dei tuoi nodi ibridi, ad esempio `topology.kubernetes.io/zone: onprem`. In alternativa, puoi impostare l'etichetta nella fase `nodeadm init`, specificando l'etichetta nella configurazione di `nodeadm`. Consultare the section called "Configurazione del nodo per personalizzare kubelet (Facoltativo)". Nota, ai nodi in esecuzione nel AWS Cloud viene applicata automaticamente un'etichetta di zona topologica che corrisponde alla zona di disponibilità (AZ) del nodo.

```
kubectl label node hybrid-node-name topology.kubernetes.io/zone=zone
```

2. Aggiungi `podAntiAffinity` all'implementazione di CoreDNS con la chiave della zona di topologia. In alternativa, puoi configurare l'implementazione di CoreDNS durante l'installazione con i componenti aggiuntivi EKS.

```
kubectl edit deployment coredns -n kube-system
```

```
spec:
  template:
    spec:
      affinity:
       ...
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: kubernetes.io/hostname
            weight: 100
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: topology.kubernetes.io/zone
            weight: 50
      ...
```

3. Aggiungi l'impostazione `trafficDistribution: PreferClose` alla configurazione del servizio `kube-dns` per abilitare la Distribuzione del traffico del servizio.

```
kubectl patch svc kube-dns -n kube-system --type=merge -p '{
  "spec": {
    "trafficDistribution": "PreferClose"
  }
```

```
}'
```

4. Puoi confermare che la Distribuzione del traffico del servizio è abilitata visualizzando le slice degli endpoint per il servizio `kube-dns`. Le sezioni degli endpoint devono mostrare gli `hints` per le etichette delle zone topologiche, a conferma che la Distribuzione del traffico del servizio è abilitata. Se non vedi l'`hints` per ciascun indirizzo degli endpoint, la Distribuzione del traffico del servizio non è abilitato.

```
kubectl get endpointslice -A | grep "kube-dns"
```

```
kubectl get endpointslice kube-dns-<id>  -n kube-system -o yaml
```

```
addressType: IPv4
apiVersion: discovery.k8s.io/v1
endpoints:
- addresses:
  - <your-hybrid-node-pod-ip>
  hints:
    forZones:
    - name: onprem
  nodeName: <your-hybrid-node-name>
  zone: onprem
- addresses:
  - <your-cloud-node-pod-ip>
  hints:
    forZones:
    - name: us-west-2a
  nodeName: <your-cloud-node-name>
  zone: us-west-2a
```

Configurazione dei webhook per i componenti aggiuntivi

I seguenti componenti aggiuntivi utilizzano i webhook e sono supportati per l'uso con nodi ibridi.

- AWS Load Balancer Controller
- CloudWatch Agente di osservabilità
- AWS Distro per OpenTelemetry (ADOT)
- `cert-manager`

Consulta le sezioni seguenti per configurare i webhook utilizzati da questi componenti aggiuntivi per l'esecuzione sui nodi in Cloud. AWS

AWS Controller del sistema di bilanciamento del carico

Per utilizzare il AWS Load Balancer Controller in una configurazione di cluster in modalità mista, è necessario eseguire il controller sui nodi in Cloud. AWS A tale scopo, aggiungi quanto segue alla configurazione dei valori di Helm o specifica i valori utilizzando la configurazione aggiuntiva EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

CloudWatch Observability Agent

L'add-on CloudWatch Observability Agent ha un operatore Kubernetes che utilizza webhook. Per eseguire l'operatore sui nodi in AWS Cloud in una configurazione di cluster in modalità mista, modifica la configurazione dell'operatore Observability Agent. CloudWatch Non puoi configurare l'affinità dell'operatore durante l'installazione con i componenti aggiuntivi Helm ed EKS (consulta problema containers-roadmap #2431).

```
kubectl edit -n amazon-cloudwatch deployment amazon-cloudwatch-observability-
controller-manager
```

```
spec:
  ...
  template:
    ...
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
```

```
                values:
                - hybrid
```

AWS Distro per OpenTelemetry (ADOT)

L'add-on AWS Distro for OpenTelemetry (ADOT) ha un operatore Kubernetes che utilizza webhook. Per eseguire l'operatore sui nodi in AWS Cloud in una configurazione del cluster in modalità mista, aggiungi quanto segue alla configurazione dei valori di Helm o specifica i valori utilizzando la configurazione del componente aggiuntivo EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Se il CIDR del pod non è instradabile sulla rete on-premises, il collettore ADOT deve essere eseguito su nodi ibridi per acquisire le metriche dai nodi ibridi e dai carichi di lavoro in esecuzione su di essi. A tale scopo, modifica la Definizione di risorse personalizzate (CRD).

```
kubectl -n opentelemetry-operator-system edit opentelemetrycollectors.opentelemetry.io
 adot-col-prom-metrics
```

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

Puoi configurare il collettore ADOT per acquisire solo le metriche dai nodi ibridi e dalle risorse in esecuzione sui nodi ibridi aggiungendo il seguente `relabel_configs` a ciascun `scrape_configs` nella configurazione CRD del collettore ADOT.

```
relabel_configs:
  - action: keep
    regex: hybrid
    source_labels:
    - __meta_kubernetes_node_label_eks_amazonaws_com_compute_type
```

L'add-on ADOT ha un requisito preliminare da installare `cert-manager` per i certificati TLS utilizzati dal webhook dell'operatore ADOT. `cert-manager`esegue anche webhook ed è possibile configurarlo per l'esecuzione su nodi in AWS Cloud con la seguente configurazione dei valori Helm.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
```

```
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

### `cert-manager`

L'`cert-manager`add-on esegue i webhook ed è possibile configurarlo per l'esecuzione su nodi in AWS Cloud con la seguente configurazione dei valori Helm.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
```

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

## Configurare il proxy per i nodi ibridi

Se utilizzi un server proxy nell'ambiente on-premises per il traffico in uscita dal data center o dall'ambiente periferico, devi configurare separatamente i nodi e il cluster per usare il server proxy.

Cluster

Sul cluster, devi configurare `kube-proxy` per utilizzare il server proxy. Devi configurare `kube-proxy` dopo aver creato il cluster Amazon EKS.

Nodi

Sui tuoi nodi, devi configurare il sistema operativo, `containerd`, `kubelet` e l'agente Amazon SSM per utilizzare il tuo server proxy. Puoi apportare queste modifiche durante il processo di creazione delle immagini del sistema operativo o prima di eseguire `nodeadm init` su ciascun nodo ibrido.

### Configurazione a livello di nodo

Devi applicare le seguenti configurazioni nelle immagini del sistema operativo o prima di eseguire `nodeadm init` su ciascun nodo ibrido.

#### Configurazione del proxy **`containerd`**

`containerd` è il runtime di gestione dei container predefinito per Kubernetes. Se utilizzi un proxy per l'accesso a Internet, devi configurare `containerd` in modo che possa estrarre le immagini dei container richieste da Kubernetes e Amazon EKS.

Crea un file su ogni nodo ibrido chiamato `http-proxy.conf` nella directory `/etc/systemd/system/containerd.service.d` con i seguenti contenuti. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configurazione di **`containerd`** a partire dai dati dell'utente

La directory `containerd.service.d` dovrà essere creata per questo file. Dovrai ricaricare systemd per recuperare il file di configurazione senza dover riavviare il sistema. In AL2023, il servizio sarà probabilmente già in esecuzione al momento dell'esecuzione dello script, quindi sarà necessario riavviarlo.

```
mkdir -p /etc/systemd/system/containerd.service.d
echo '[Service]' > /etc/systemd/system/containerd.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/containerd.service.d/
http-proxy.conf
systemctl daemon-reload
systemctl restart containerd
```

Configurazione del proxy **`kubelet`**

`kubelet` è l'agente del nodo Kubernetes che viene eseguito su ogni nodo Kubernetes ed è responsabile della gestione del nodo e dei pod in esecuzione su di esso. Se utilizzi un proxy nel tuo ambiente on-premises, devi configurare `kubelet` in modo che possa comunicare con gli endpoint pubblici o privati del cluster Amazon EKS.

Crea un file su ogni nodo ibrido chiamato `http-proxy.conf` nella directory `/etc/systemd/system/kubelet.service.d/` col seguente contenuto. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configurazione di **`kubelet`** a partire dai dati dell'utente

La directory `kubelet.service.d` deve essere creata per questo file. Dovrai ricaricare systemd per recuperare il file di configurazione senza dover riavviare il sistema. In AL2023, il servizio sarà probabilmente già in esecuzione al momento dell'esecuzione dello script, quindi sarà necessario riavviarlo.

```
mkdir -p /etc/systemd/system/kubelet.service.d
echo '[Service]' > /etc/systemd/system/kubelet.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/kubelet.service.d/http-
proxy.conf
systemctl daemon-reload
systemctl restart kubelet
```

Configurazione del proxy **`ssm`**

`ssm` è uno dei provider di credenziali che possono essere utilizzati per inizializzare un nodo ibrido. `ssm` è responsabile dell'autenticazione con AWS e della generazione di credenziali temporanee utilizzate da `kubelet`. Se utilizzi un proxy nel tuo ambiente on-premises e usi `ssm` come fornitore di credenziali sul nodo, devi configurare `ssm` in modo che possa comunicare con gli endpoint del servizio Amazon SSM.

Crea un file su ogni nodo ibrido chiamato `http-proxy.conf` nel percorso seguente, a seconda del sistema operativo

- Ubuntu: `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d/http-proxy.conf`
- Amazon Linux 2023 e Red Hat Enterprise Linux: `/etc/systemd/system/amazon-ssm-agent.service.d/http-proxy.conf`

Popola il file con i seguenti contenuti. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
```

```
Environment="NO_PROXY=localhost"
```

Configurazione di **`ssm`** a partire dai dati dell'utente

La directory `ssm` del file di servizio systemd deve essere creata per questo file. Il percorso della directory dipende dal sistema operativo utilizzato nel nodo.

- Ubuntu: `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d`
- Amazon Linux 2023 e Red Hat Enterprise Linux: `/etc/systemd/system/amazon-ssm-agent.service.d`

Sostituisci il nome del servizio systemd nel comando riavvio riportato di seguito a seconda del sistema operativo utilizzato nel nodo

- Ubuntu: `snap.amazon-ssm-agent.amazon-ssm-agent`
- Amazon Linux 2023 e Red Hat Enterprise Linux: `amazon-ssm-agent`

```
mkdir -p systemd-service-file-directory
echo '[Service]' > [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://[.replaceable]#proxy-domain:port"' >> systemd-
service-file-directory/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://[.replaceable]#proxy-domain:port"' >>
 [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> [.replaceable]#systemd-service-file-
directory/http-proxy.conf
systemctl daemon-reload
systemctl restart [.replaceable]#systemd-service-name
```

Configurazione del proxy del sistema operativo

Se utilizzi un proxy per l'accesso a Internet, devi configurare il sistema operativo in modo da poter estrarre le dipendenze dei nodi ibridi dal gestore di pacchetti dei sistemi operativi.

Ubuntu

1. Configura `snap` per utilizzare il proxy con i seguenti comandi:

```
sudo snap set system proxy.https=http://proxy-domain:port
```

```
sudo snap set system proxy.http=http://proxy-domain:port
```

2. Per abilitare il proxy per `apt`, crea un file chiamato `apt.conf` nella directory `/etc/apt/`. Sostituisci proxy-domain e port con i valori per il tuo ambiente.

```
Acquire::http::Proxy "http://proxy-domain:port";
Acquire::https::Proxy "http://proxy-domain:port";
```

Amazon Linux 2023

1. Configura `dnf` per l'utilizzo del proxy. Crea un file `/etc/dnf/dnf.conf` con i valori proxy-domain e port per il tuo ambiente.

```
proxy=http://proxy-domain:port
```

Red Hat Enterprise Linux

1. Configura `yum` per l'utilizzo del proxy. Crea un file `/etc/yum.conf` con i valori proxy-domain e port per il tuo ambiente.

```
proxy=http://proxy-domain:port
```

Configurazione del proxy di IAM Roles Anywhere

Il servizio del fornitore di credenziali IAM Roles Anywhere è responsabile dell'aggiornamento delle credenziali quando si utilizza IAM Roles Anywhere con il flag `enableCredentialsFile` (consulta the section called "EKS Pod Identity Agent"). Se utilizzi un proxy nel tuo ambiente on-premises, devi configurare il servizio in modo che possa comunicare con gli endpoint di IAM Roles Anywhere.

Crea un file denominato `http-proxy.conf` nella directory `/etc/systemd/system/aws_signing_helper_update.service.d/` con i seguenti contenuti. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
```

```
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configurazione a livello di cluster

Le configurazioni in questa sezione devono essere applicate dopo aver creato il cluster Amazon EKS e prima di eseguire `nodeadm init` su ogni nodo ibrido.

Configurazione di un proxy kube-proxy

Amazon EKS installa automaticamente `kube-proxy` su ogni nodo ibrido come DaemonSet quando i nodi entrano a far parte del cluster. `kube-proxy` abilita il routing tra servizi supportati da pod su cluster Amazon EKS. Per configurare ogni host, `kube-proxy` necessita della risoluzione DNS per l'endpoint del cluster Amazon EKS.

1. Modifica il DaemonSet `kube-proxy` mediante il comando seguente

   ```
   kubectl -n kube-system edit ds kube-proxy
   ```

   Verrà aperta la definizione del DaemonSet `kube-proxy` sull'editor configurato.

2. Aggiungi le variabili di ambiente per `HTTP_PROXY` e `HTTPS_PROXY`. Nota che la variabile di ambiente `NODE_NAME` dovrebbe già esistere nella configurazione. Sostituisci `proxy-domain` e `port` con valori per il tuo ambiente.

   ```
   containers:
     - command:
       - kube-proxy
       - --v=2
       - --config=/var/lib/kube-proxy-config/config - --hostname-override=$(NODE_NAME)
       env:
       - name: HTTP_PROXY
         value: http://proxy-domain:port
       - name: HTTPS_PROXY
         value: http://proxy-domain:port
       - name: NODE_NAME
         valueFrom:
           fieldRef:
             apiVersion: v1
             fieldPath: spec.nodeName
   ```

# Configurazione di Cilium BGP per nodi ibridi

Questo argomento descrive come configurare il Border Gateway Protocol (BGP) di Cilium per Amazon EKS Hybrid Nodes. La funzionalità BGP di Cilium si chiama Cilium BGP Control Plane e può essere utilizzata per pubblicizzare indirizzi di pod e servizi sulla rete on-premises. Per metodi alternativi per rendere i pod CIDR instradabili sulla on-premises, consulta the section called "CIDR dei pod remoti instradabili".

## Configurazione del BGP di Cilium

### Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".

### Procedura

1. Per utilizzare BGP con Cilium per pubblicizzare pod o indirizzi di servizi sulla rete on-premises, è necessario installare Cilium con `bgpControlPlane.enabled: true`. Se stai abilitando BGP per un'implementazione Cilium esistente, devi riavviare l'operatore Cilium per applicare la configurazione BGP se BGP non era stato precedentemente abilitato. È possibile impostare i valori di Helm `operator.rollOutPods` per `true` al fine di riavviare l'operatore Cilium come parte del processo di installazione/aggiornamento di Helm.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --set bgpControlPlane.enabled=true
```

2. Conferma che l'operatore e gli agenti Cilium sono stati riavviati e sono in esecuzione.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
```

```
cilium-z7hlb                       1/1     Running   0          6m30s
```

3. Crea un file chiamato `cilium-bgp-cluster.yaml` con una definizione `CiliumBGPClusterConfig`. Potrebbe essere necessario ottenere le seguenti informazioni dall'amministratore di rete.
   - Configura `localASN` con l'ASN per i nodi che eseguono Cilium.
   - Configura `peerASN` con l'ASN per il router on-premises.
   - Configura `peerAddress` con l'IP del router on-premises con cui verrà eseguito il peering di ogni nodo su cui è in esecuzione Cilium.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPClusterConfig
metadata:
  name: cilium-bgp
spec:
  nodeSelector:
    matchExpressions:
    - key: eks.amazonaws.com/compute-type
      operator: In
      values:
      - hybrid
  bgpInstances:
  - name: "rack0"
    localASN: NODES_ASN
    peers:
    - name: "onprem-router"
      peerASN: ONPREM_ROUTER_ASN
      peerAddress: ONPREM_ROUTER_IP
      peerConfigRef:
        name: "cilium-peer"
```

4. Applica la configurazione del cluster Cilium BGP al cluster.

```
kubectl apply -f cilium-bgp-cluster.yaml
```

5. Crea un file denominato `cilium-bgp-peer.yaml` con la risorsa `CiliumBGPPeerConfig` che definisce una configurazione peer BGP. Più peer possono condividere la stessa configurazione e fornire un riferimento alla risorsa comune `CiliumBGPPeerConfig`. Consulta BGP Peer configuration nella documentazione di Cilium per un elenco completo delle opzioni di configurazione.

I valori per le seguenti impostazioni peer Cilium devono corrispondere a quelli del router on-premises con cui si esegue il peering.

- Configura `holdTimeSeconds` che determina per quanto tempo un peer BGP attende un messaggio keepalive o update prima di dichiarare la sessione inattiva. Il valore predefinito è 90 secondi.
- Configura `keepAliveTimeSeconds` che determina se un peer BGP è ancora raggiungibile e la sessione BGP è attiva. Il valore predefinito è 30 secondi.
- Configura `restartTimeSeconds` che determina l'ora in cui il piano di controllo BGP di Cilium dovrebbe ristabilire la sessione BGP dopo un riavvio. Il valore predefinito è 120 secondi.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPPeerConfig
metadata:
  name: cilium-peer
spec:
  timers:
    holdTimeSeconds: 90
    keepAliveTimeSeconds: 30
  gracefulRestart:
    enabled: true
    restartTimeSeconds: 120
  families:
    - afi: ipv4
      safi: unicast
      advertisements:
        matchLabels:
          advertise: "bgp"
```

6. Applica la configurazione del peer Cilium BGP al cluster.

```
kubectl apply -f cilium-bgp-peer.yaml
```

7. Crea un file denominato `cilium-bgp-advertisement-pods.yaml` con una risorsa `CiliumBGPAdvertisement` per pubblicizzare i pod CIDR sulla rete on-premises.
   - La risorsa `CiliumBGPAdvertisement` viene utilizzata per definire i tipi di pubblicità e gli attributi ad essi associati. L'esempio seguente configura Cilium per pubblicizzare solo i pod CIDR. Vedi gli esempi in the section called "Tipo di servizio LoadBalancer" e the section called "Bilanciamento del carico Cilium all'interno del cluster" per ulteriori informazioni sulla configurazione di Cilium per pubblicizzare gli indirizzi dei servizi.

- Ogni nodo ibrido che esegue l'agente Cilium esegue il peering con il router upstream abilitato per BGP. Ogni nodo pubblicizza la gamma di pod CIDR che possiede quando `advertisementType` di Cilium è impostato su `PodCIDR` come nell'esempio seguente. Per ulteriori informazioni, consulta BGP Advertisements configuration nella documentazione di Cilium.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-pods
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "PodCIDR"
```

8. Applica la configurazione degli annunci pubblicitari BGP di Cilium al cluster.

```
kubectl apply -f cilium-bgp-advertisement-pods.yaml
```

9. È possibile confermare che il peering BGP ha funzionato con la CLI di Cilium utilizzando il comando `cilium bgp peers`. Dovresti vedere i valori corretti nell'output per il tuo ambiente e lo stato della sessione come `established`. Per ulteriori informazioni sulla risoluzione dei problemi, consulta Troubleshooting and Operations Guide nella documentazione di Cilium.

   Negli esempi seguenti, ci sono cinque nodi ibridi che eseguono l'agente Cilium e ogni nodo pubblicizza la gamma Pod CIDR di sua proprietà.

```
cilium bgp peers
```

```
Node                   Local AS    Peer AS                Peer Address        Session
 State   Uptime     Family         Received   Advertised
mi-026d6a261e355fba7   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h18m58s   ipv4/unicast   1
      2
mi-082f73826a163626e   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h19m12s   ipv4/unicast   1
      2
```

```
mi-09183e8a3d755abf6   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h18m47s   ipv4/unicast   1
      2
mi-0d78d815980ed202d   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h19m12s   ipv4/unicast   1
      2
mi-0daa253999fe92daa   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h18m58s   ipv4/unicast   1
      2
```

```
cilium bgp routes
```

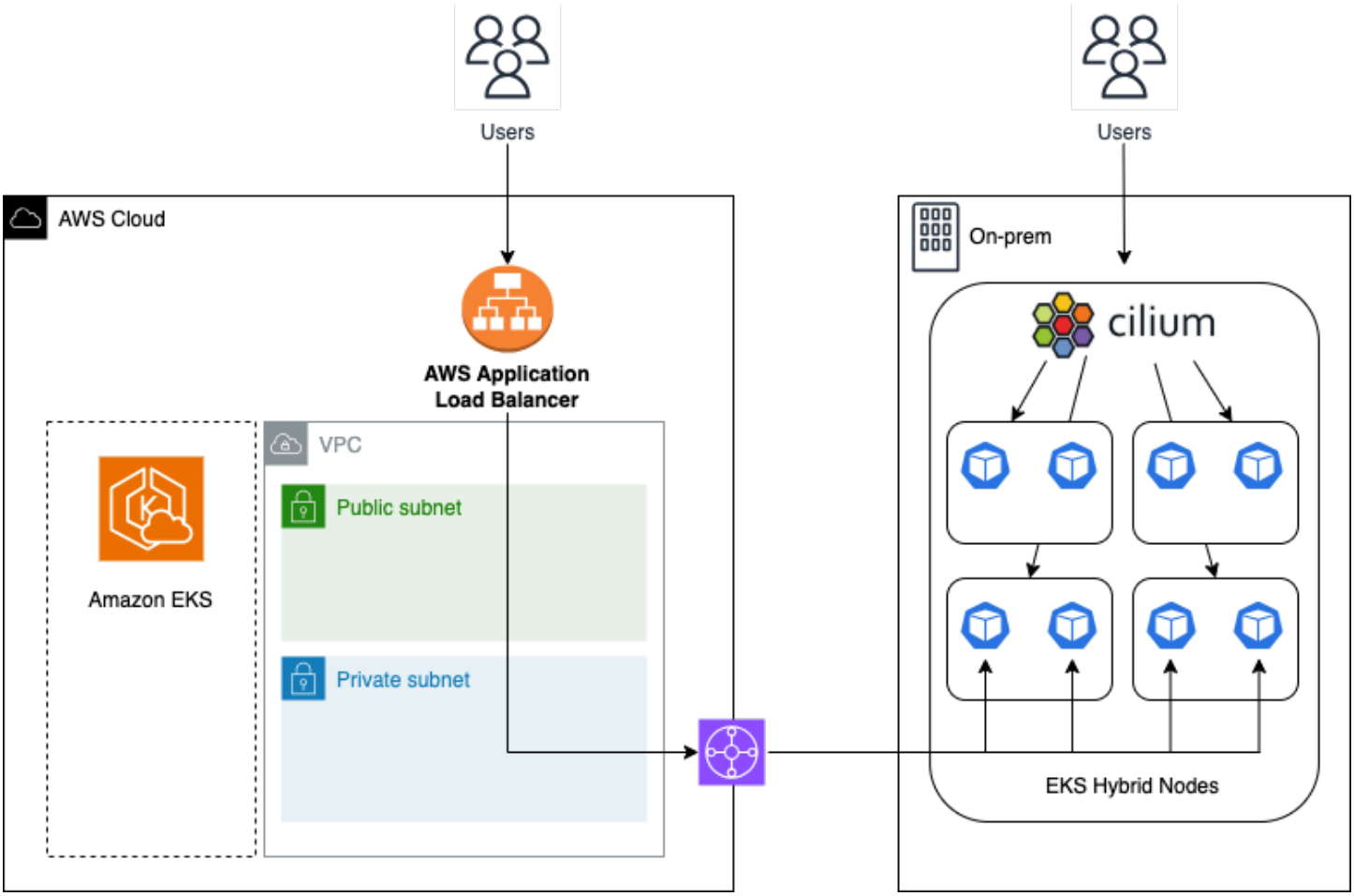
```
Node                   VRouter       Prefix           NextHop   Age         Attrs
mi-026d6a261e355fba7   NODES_ASN     10.86.2.0/26     0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-082f73826a163626e   NODES_ASN     10.86.2.192/26   0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-09183e8a3d755abf6   NODES_ASN     10.86.2.64/26    0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-0d78d815980ed202d   NODES_ASN     10.86.2.128/26   0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-0daa253999fe92daa   NODES_ASN     10.86.3.0/26     0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
```

## Configurazione dell'ingresso Kubernetes per nodi ibridi

Questo argomento descrive come configurare l'ingresso Kubernetes per i carichi di lavoro in esecuzione su Amazon EKS Hybrid Nodes. L'ingresso Kubernetes espone i percorsi HTTP e HTTPS dall'esterno del cluster ai servizi all'interno del cluster. Per utilizzare le risorse dell'ingresso, è necessario un controller dell'ingresso Kubernetes per configurare l'infrastruttura di rete e i componenti che servono il traffico di rete.

AWS supporta AWS Application Load Balancer (ALB) e Cilium for Kubernetes Ingress per carichi di lavoro in esecuzione su nodi ibridi EKS. La decisione di utilizzare ALB o Cilium per l'ingresso si basa sulla fonte del traffico delle applicazioni. Se il traffico dell'applicazione proviene da una AWS regione, AWS consiglia di utilizzare AWS ALB e il Load AWS Balancer Controller. Se il traffico delle

applicazioni proviene dall'ambiente locale locale locale o perimetrale, AWS consiglia di utilizzare le funzionalità Ingress integrate di Cilium, che possono essere utilizzate con o senza l'infrastruttura di bilanciamento del carico nell'ambiente.




AWS Application Load Balancer

Puoi utilizzare il Controller del bilanciatore del carico AWS e Application Load Balancer (ALB) con il tipo di destinazione `ip` per i carichi di lavoro in esecuzione su nodi ibridi. Quando si utilizza il tipo di destinazione `ip`, ALB inoltra il traffico direttamente ai pod, aggirando il percorso di rete del livello di servizio. Affinché ALB raggiunga le destinazioni IP del pod sui nodi ibridi, il CIDR del pod on-premises deve essere instradabile sulla rete locale. Inoltre, il AWS Load Balancer Controller utilizza webhook e richiede una comunicazione diretta dal piano di controllo EKS. Per ulteriori informazioni, consulta the section called "Configurazione di webhook".

### Considerazioni

- the section called "Installa con Helm"Per ulteriori informazioni, vedere the section called "Bilanciamento del carico di applicazione" AWS Application Load Balancer e Load Balancer AWS Controller.
- Consulta Best Practices for Load Balancing per informazioni su come scegliere tra AWS Application Load Balancer AWS e Network Load Balancer.
- Vedi AWS Load Balancer Controller Ingress annotations per l'elenco delle annotazioni che possono essere configurate per le risorse Ingress con Application Load Balancer. AWS

### Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non si desidera utilizzare BGP, è necessario utilizzare un metodo alternativo per rendere il pod locale instradabile sulla rete locale. CIDRs Se non rendi CIDRs instradabile il pod locale, ALB non sarà in grado di registrarlo o contattare i destinatari IP del pod.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- eksctl è installato nell'ambiente a riga di comando, consulta le istruzioni di installazione di eksctl per ulteriori informazioni.

### Procedura

1. Scarica una policy IAM per il AWS Load Balancer Controller che gli consenta di effettuare chiamate per tuo AWS APIs conto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Sostituisci il valore per cluster name (`CLUSTER_NAME`), AWS Region (`AWS_REGION`) e AWS account ID (`AWS_ACCOUNT_ID`) con le tue impostazioni ed esegui il comando seguente.

```
eksctl create iamserviceaccount \
    --cluster=CLUSTER_NAME \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws:iam::AWS_ACCOUNT_ID:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region AWS_REGION \
    --approve
```

4. Aggiungi il repository di grafici Helm di eks-charts e aggiorna il repository Helm locale per assicurarti di avere i grafici più recenti.

```
helm repo add eks https://aws.github.io/eks-charts
```

```
helm repo update eks
```

5. Installa il AWS Load Balancer Controller. Sostituisci il valore per cluster name (`CLUSTER_NAME`), AWS Region (`AWS_REGION`), VPC ID (`VPC_ID`) e Load AWS Balancer Controller Helm chart version `AWS_LBC_HELM_VERSION` () con le tue impostazioni ed esegui il comando seguente. Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi in AWS Cloud, puoi eseguire il AWS Load Balancer Controller sui nodi cloud seguendo le istruzioni riportate all'indirizzo. the section called "AWS Controller del sistema di bilanciamento del carico"

   - Puoi trovare la versione più recente del grafico Helm eseguendo `helm search repo eks/aws-load-balancer-controller --versions`.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --version AWS_LBC_HELM_VERSION \
  --set clusterName=CLUSTER_NAME \
  --set region=AWS_REGION \
  --set vpcId=VPC_ID \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

6. Verifica che il AWS Load Balancer Controller sia stato installato correttamente.

```
kubectl get -n kube-system deployment aws-load-balancer-controller
```

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

7. Crea un'applicazione di esempio. L'esempio seguente utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

8. Crea un file denominato `my-ingress-alb.yaml` con i seguenti contenuti.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
  annotations:
    alb.ingress.kubernetes.io/load-balancer-name: "my-ingress-alb"
    alb.ingress.kubernetes.io/target-type: "ip"
    alb.ingress.kubernetes.io/scheme: "internet-facing"
    alb.ingress.kubernetes.io/healthcheck-path: "/details/1"
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

9. Applica la configurazione dell'ingresso al cluster.

```
kubectl apply -f my-ingress-alb.yaml
```

10. Il provisioning dell'ALB per la risorsa in ingresso può richiedere alcuni minuti. Una volta effettuato il provisioning dell'ALB, alla risorsa Ingress verrà assegnato un indirizzo che corrisponde al nome DNS dell'implementazione ALB. L'indirizzo avrà il formato `<alb-name>-<random-string>.<region>.elb.amazonaws.com`.

```
kubectl get ingress my-ingress
```

```
NAME         CLASS   HOSTS   ADDRESS
    PORTS   AGE
my-ingress   alb     *       my-ingress-alb-<random-
string>.<region>.elb.amazonaws.com   80      23m
```

11. Accedi al Servizio utilizzando l'indirizzo dell'ALB.

```
curl -s http//my-ingress-alb-<random-string>.<region>.elb.amazonaws.com:80/details/1
 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
  "details": "This is the details page"
}
```

## Panoramica dell'ingresso di Cilium e del gateway di Cilium

Le funzionalità dell'ingresso di Cilium sono integrate nell'architettura di Cilium e possono essere gestite con l'API dell'ingresso di Kubernetes o l'API Gateway. Se non disponi di risorse Ingress esistenti, ti consigliamo di AWS iniziare con l'API Gateway, poiché è un modo più espressivo e flessibile per definire e gestire le risorse di rete Kubernetes. L'API del Gateway di Kubernetes mira a standardizzare il modo in cui le risorse di rete per ingresso, bilanciamento del carico e mesh di servizi vengono definite e gestite nei cluster Kubernetes.

Quando abiliti le funzionalità di ingresso o gateway di Cilium, l'operatore Cilium riconcilia gli oggetti ingresso/gateway nel cluster e i proxy Envoy su ciascun nodo elaborano il traffico di rete di livello 7 (L7). Cilium non fornisce direttamente l'infrastruttura ingresso/gateway come i bilanciatori del carico. Se prevedi di utilizzare l'ingresso/gateway di Cilium con un bilanciatore del carico, devi utilizzare gli strumenti di quest'ultimo, in genere un controller di ingresso o gateway, per distribuire e gestire l'infrastruttura del sistema del bilanciatore del carico.

Per il traffico ingresso/gateway, Cilium gestisce il traffico di rete principale e l'applicazione delle policy L3/L4, mentre i proxy Envoy integrati elaborano il traffico di rete L7. Con l'ingresso/gateway di Cilium, Envoy è responsabile dell'applicazione delle regole di routing L7, delle policy e della manipolazione delle richieste, della gestione avanzata del traffico come la suddivisione e il mirroring del traffico e della terminazione e origine del TLS. I proxy Envoy di Cilium vengono distribuiti come proxy separati DaemonSet (`cilium-envoy`) per impostazione predefinita, il che consente a Envoy e all'agente Cilium di essere aggiornati, scalati e gestiti separatamente.

Per ulteriori informazioni su come funzionano l'ingresso di Cilium e il gateway di Cilium, consulta le pagine Cilium Ingress e Cilium Gateway nella documentazione di Cilium.

Confronto tra ingresso e gateway di Cilium

La tabella seguente riassume le funzionalità di ingresso e gateway di Cilium, a partire dalla versione 1.17.x di Cilium.

| Funzionalità | Ingresso | Gateway |
|---|---|---|
| Tipo di servizio LoadBalancer | Sì | Sì |
| Tipo di servizio NodePort | Sì | No[1] |
| Rete host | Sì | Sì |
| Bilanciatore del carico condiviso | Sì | Sì |
| Bilanciatore del carico dedicato | Sì | No[2] |
| Policy di rete | Sì | Sì |
| Protocolli | Livello 7 (HTTP(S), gRPC) | Livello 7 (HTTP(S), gRPC)[3] |

| Funzionalità | Ingresso | Gateway |
| --- | --- | --- |
| TLS Passthrough | Sì | Sì |
| Gestione del traffico | Routing del percorso e dell'host | Routing del percorso e dell'host, reindirizzamento e riscrittura degli URL, suddivisione del traffico, modifica dell'intestazione |

[1] Il supporto per i NodePort servizi di Cilium Gateway è previsto per la versione 1.18.x di Cilium (#27273)

[2] Supporto del gateway di Cilium per bilanciatori del carico dedicati (#25567)

[3] Supporto Cilium Gateway per (#21929) TCP/UDP

Installazione del gateway di Cilium

Considerazioni

- Cilium deve essere configurato con `nodePort.enabled` impostato su `true` come mostrato negli esempi seguenti. Se utilizzi la funzionalità di sostituzione kube-proxy di Cilium, non devi impostare `nodePort.enabled` su `true`.
- Cilium deve essere configurato con `envoy.enabled` impostato su `true` come mostrato negli esempi seguenti.
- Il gateway di Cilium può essere distribuito nel bilanciatore del carico (impostazione predefinita) o in modalità rete host.
- Quando si utilizza Cilium Gateway in modalità load balancer, l'`service.beta.kubernetes.io/aws-load-balancer-type: "external"`annotazione deve essere impostata sulla risorsa Gateway per impedire al provider AWS cloud legacy di creare un Classic Load Balancer for the Service del LoadBalancer tipo che Cilium crea per la risorsa Gateway.
- Quando si utilizza Cilium Gateway in modalità rete host, la modalità Service of type è disabilitata. LoadBalancer La modalità di rete host è utile per gli ambienti che non dispongono di un'infrastruttura del bilanciatore del carico; per ulteriori informazioni, consulta the section called "Rete host".

Prerequisiti

1. Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.
2. Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".

Procedura

1. Installa le definizioni delle risorse personalizzate dell'API Kubernetes Gateway (). CRDs

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_gatewayclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_gateways.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_httproutes.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_referencegrants.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_grpcroutes.yaml
```

2. Crea un file denominato `cilium-gateway-values.yaml` con i seguenti contenuti. L'esempio seguente configura Cilium Gateway per utilizzare la modalità di bilanciamento del carico predefinita e per utilizzare un proxy Envoy separato configurato `cilium-envoy` DaemonSet per l'esecuzione solo su nodi ibridi.

```
gatewayAPI:
  enabled: true
  # uncomment to use host network mode
  # hostNetwork:
  #   enabled: true
nodePort:
  enabled: true
envoy:
  enabled: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
```

```
            operator: In
            values:
            - hybrid
```

3. Applica il file di valori Helm al cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --values cilium-gateway-values.yaml
```

4. Verifica che l'operatore, l'agente e i pod Envoy Cilium siano in esecuzione.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-envoy-5pgnd                 1/1     Running   0          6m31s
cilium-envoy-6fhg4                 1/1     Running   0          6m30s
cilium-envoy-jskrk                 1/1     Running   0          6m30s
cilium-envoy-k2xtb                 1/1     Running   0          6m31s
cilium-envoy-w5s9j                 1/1     Running   0          6m31s
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

### Configurazione del gateway di Cilium

Il gateway di Cilium è abilitato sugli oggetti del gateway impostando `gatewayClassName` su `cilium`. Il servizio creato da Cilium per le risorse del gateway può essere configurato con campi sull'oggetto del gateway. Le annotazioni comuni utilizzate dai controller del gateway per configurare l'infrastruttura del bilanciatore del carico possono essere configurate con il campo `infrastructure` dell'oggetto del gateway. Quando si utilizza l' LoadBalancer IPAM di Cilium (vedi esempio inthe section called "Tipo di servizio LoadBalancer"), l'indirizzo IP da utilizzare per il Service of type LoadBalancer può essere configurato nel campo dell'oggetto Gateway. `addresses` Per ulteriori informazioni sulla configurazione del gateway, consulta Kubernetes Gateway API specification.

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  infrastructure:
    annotations:
      service.beta.kubernetes.io/...
      service.kubernetes.io/...
  addresses:
  - type: IPAddress
    value: <LoadBalancer IP address>
  listeners:
  ...
```

Cilium e le specifiche Kubernetes Gateway supportano, Gateway GatewayClass, e le risorse. HTTPRoute GRPCRoute ReferenceGrant

- Vedi HTTPRoutee GRPCRoutespecifiche per l'elenco dei campi disponibili.
- Vedi gli esempi nella sezione the section called "Implementazione del gateway di Cilium" seguente e gli esempi in Cilium documentation per scoprire come utilizzare e configurare queste risorse.

Implementazione del gateway di Cilium

1. Crea un'applicazione di esempio. L'esempio seguente utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

   ```
   kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
   samples/bookinfo/platform/kube/bookinfo.yaml
   ```

2. Verifica che l'applicazione sia in esecuzione correttamente.

   ```
   kubectl get pods
   ```

   ```
   NAME                              READY   STATUS    RESTARTS   AGE
   details-v1-766844796b-9965p       1/1     Running   0          81s
   productpage-v1-54bb874995-jmc8j   1/1     Running   0          80s
   ratings-v1-5dc79b6bcd-smzxz       1/1     Running   0          80s
   reviews-v1-598b896c9d-vj7gb       1/1     Running   0          80s
   ```

```
reviews-v2-556d6457d-xbt8v        1/1     Running   0          80s
reviews-v3-564544b4d6-cpmvq       1/1     Running   0          80s
```

3. Crea un file denominato `my-gateway.yaml` con i seguenti contenuti. L'esempio seguente utilizza l'`service.beta.kubernetes.io/aws-load-balancer-type: "external"`annotazione per impedire al provider AWS cloud legacy di creare un Classic Load Balancer per il servizio del LoadBalancer tipo creato da Cilium per la risorsa Gateway.

```
---
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  infrastructure:
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-type: "external"
  listeners:
  - protocol: HTTP
    port: 80
    name: web-gw
    allowedRoutes:
      namespaces:
        from: Same
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /details
    backendRefs:
    - name: details
      port: 9080
```

4. Applica la risorsa del gateway al cluster.

```
kubectl apply -f my-gateway.yaml
```

5. Conferma che la risorsa del gateway e il servizio corrispondente sono stati creati. In questa fase, si prevede che il ADDRESS campo della risorsa Gateway non sia popolato con un indirizzo IP o un nome host e che al servizio di tipo LoadBalancer relativo alla risorsa Gateway non sia assegnato un indirizzo IP o un nome host.

```
kubectl get gateway my-gateway
```

```
NAME         CLASS    ADDRESS   PROGRAMMED   AGE
my-gateway   cilium             True         10s
```

```
kubectl get svc cilium-gateway-my-gateway
```

```
NAME                         TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
cilium-gateway-my-gateway   LoadBalancer   172.16.227.247   <pending>     80:30912/
TCP   24s
```

6. Procedi the section called "Tipo di servizio LoadBalancer" alla configurazione della risorsa Gateway per utilizzare un indirizzo IP allocato da Cilium Load Balancer IPAM the section called "Rete host" e/o alla configurazione della risorsa Gateway per NodePort utilizzare o ospitare indirizzi di rete. the section called "Tipo di servizio NodePort"

Installazione dell'ingresso di Cilium

Considerazioni

- Cilium deve essere configurato con `nodePort.enabled` impostato su `true` come mostrato negli esempi seguenti. Se utilizzi la funzionalità di sostituzione kube-proxy di Cilium, non devi impostare `nodePort.enabled` su `true`.
- Cilium deve essere configurato con `envoy.enabled` impostato su `true` come mostrato negli esempi seguenti.
- Con `ingressController.loadbalancerMode` impostato su `dedicated`, Cilium crea servizi dedicati per ogni risorsa dell'ingresso. Con `ingressController.loadbalancerMode` set

`toshared`, Cilium crea un servizio condiviso di tipo LoadBalancer per tutte le risorse Ingress nel cluster. Quando si utilizza la modalità del bilanciatore del carico `shared`, le impostazioni per il servizio condiviso come `labels`, `annotations`, `type` e `loadBalancerIP` sono configurate nella sezione dei valori Helm `ingressController.service`. Per ulteriori informazioni, consulta Cilium Helm values reference.

- Con `ingressController.default` impostato su `true`, Cilium è configurato come controller di ingresso predefinito per il cluster e creerà voci di ingresso anche quando `ingressClassName` non è specificato nelle risorse di ingresso.
- L'ingresso di Cilium può essere distribuito nel bilanciatore del carico (impostazione predefinita), nella porta del nodo o in modalità rete host. Quando Cilium è installato in modalità rete host, le modalità Servizio di tipo LoadBalancer e Servizio di tipo NodePort sono disabilitate. Per ulteriori informazioni, consulta the section called "Rete host".
- Imposta sempre su `ingressController.service.annotations service.beta.kubernetes.io/aws-load-balancer-type: "external"` nei valori Helm per impedire al provider AWS cloud legacy di creare un Classic Load Balancer per il servizio `cilium-ingress` predefinito creato dal grafico Cilium Helm.

Prerequisiti

1. Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.
2. Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".

Procedura

1. Crea un file denominato `cilium-ingress-values.yaml` con i seguenti contenuti. L'esempio seguente configura Cilium Ingress per utilizzare la `dedicated` modalità di bilanciamento del carico predefinita e per utilizzare un proxy separato `cilium-envoy` DaemonSet per Envoy configurato per l'esecuzione solo su nodi ibridi.

```
ingressController:
  enabled: true
  loadbalancerMode: dedicated
  service:
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-type: "external"
nodePort:
```

```
  enabled: true
envoy:
  enabled: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

2. Applica il file di valori Helm al cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --values cilium-ingress-values.yaml
```

3. Verifica che l'operatore, l'agente e i pod Envoy Cilium siano in esecuzione.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-envoy-5pgnd                 1/1     Running   0          6m31s
cilium-envoy-6fhg4                 1/1     Running   0          6m30s
cilium-envoy-jskrk                 1/1     Running   0          6m30s
cilium-envoy-k2xtb                 1/1     Running   0          6m31s
cilium-envoy-w5s9j                 1/1     Running   0          6m31s
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

Configurazione dell'ingresso di Cilium

L'ingresso di Cilium è abilitato sugli oggetti dell'ingresso impostando `ingressClassName` su `cilium`. I servizi che Cilium crea per le risorse dell'ingresso possono essere configurati con annotazioni sugli oggetti dell'ingresso quando si utilizza la modalità del bilanciatore del carico `dedicated` e nella configurazione Cilium/Helm quando si utilizza la modalità del bilanciatore del carico `shared`. Queste annotazioni vengono comunemente utilizzate dai controller dell'ingresso per configurare l'infrastruttura del bilanciatore del carico o altri attributi del Servizio come il tipo di servizio, la modalità del bilanciatore del carico, le porte e il passthrough TLS. Le annotazioni chiave sono descritte di seguito. Per un elenco completo delle annotazioni supportate, consulta Cilium Ingress annotations nella documentazione di Cilium.

| Annotazione | Description |
|---|---|
| `ingress.cilium.io/loadbalancer-mode` | `dedicated` : Servizio dedicato di tipo LoadBalancer per ogni risorsa Ingress (impostazione predefinita).<br><br>`shared`: Servizio unico di tipo LoadBalancer per tutte le risorse Ingress. |
| `ingress.cilium.io/service-type` | `LoadBalancer` : Il servizio sarà di tipo LoadBalancer (impostazione predefinita)<br><br>`NodePort`: Il servizio sarà di tipo NodePort. |
| `service.beta.kubernetes.io/aws-load-balancer-type` | `"external"` : Impedisci ai provider di AWS cloud legacy di effettuare il provisioning di Classic Load Balancer for Services di tipo Classic Load Balancer for Services. LoadBalancer |
| `lbipam.cilium.io/ips` | Elenco di indirizzi IP da allocare da Cilium IPAM LoadBalancer |

Cilium e le specifiche dell'ingresso di Kubernetes supportano regole di corrispondenza specifiche per Exact, Prefix e Implementation per i percorsi dell'ingresso. Cilium supporta l'espressione regolare come regola di corrispondenza specifica per l'implementazione. Per ulteriori informazioni, consulta Ingress path types and precedence e Path types examples nella documentazione di Cilium e gli esempi nella sezione the section called "Implementazione dell'ingresso di Cilium" di questa pagina.

Un esempio di oggetto dell'ingresso di Cilium è mostrato di seguito.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    service.beta.kubernetes.io/...
    service.kubernetes.io/...
spec:
  ingressClassName: cilium
  rules:
  ...
```

Implementazione dell'ingresso di Cilium

1. Crea un'applicazione di esempio. L'esempio seguente utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

   ```
   kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
   samples/bookinfo/platform/kube/bookinfo.yaml
   ```

2. Verifica che l'applicazione sia in esecuzione correttamente.

   ```
   kubectl get pods
   ```

   ```
   NAME                              READY   STATUS    RESTARTS   AGE
   details-v1-766844796b-9965p       1/1     Running   0          81s
   productpage-v1-54bb874995-jmc8j   1/1     Running   0          80s
   ratings-v1-5dc79b6bcd-smzxz       1/1     Running   0          80s
   reviews-v1-598b896c9d-vj7gb       1/1     Running   0          80s
   reviews-v2-556d6457d-xbt8v        1/1     Running   0          80s
   reviews-v3-564544b4d6-cpmvq       1/1     Running   0          80s
   ```

3. Crea un file denominato `my-ingress.yaml` con i seguenti contenuti. L'esempio seguente utilizza l'`service.beta.kubernetes.io/aws-load-balancer-type: "external"`annotazione per impedire al provider AWS cloud legacy di creare un Classic Load Balancer per il servizio del LoadBalancer tipo creato da Cilium per la risorsa Ingress.

   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   ```

```
metadata:
  name: my-ingress
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
spec:
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

4. Applica la risorsa dell'ingresso al cluster.

```
kubectl apply -f my-ingress.yaml
```

5. Conferma che la risorsa dell'ingresso e il servizio corrispondente sono stati creati. In questa fase, si prevede che il ADDRESS campo della risorsa Ingress non sia popolato con un indirizzo IP o un nome host e che analogamente al servizio condiviso o dedicato LoadBalancer per la risorsa Ingress non sia assegnato un indirizzo IP o un nome host.

```
kubectl get ingress my-ingress
```

```
NAME         CLASS    HOSTS   ADDRESS   PORTS   AGE
my-ingress   cilium   *                 80      8s
```

Per la modalità del bilanciatore del carico `shared`

```
kubectl -n kube-system get svc cilium-ingress
```

```
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
      AGE
cilium-ingress   LoadBalancer   172.16.217.48   <pending>     80:32359/TCP,443:31090/
TCP   10m
```

Per la modalità del bilanciatore del carico `dedicated`

```
kubectl -n default get svc cilium-ingress-my-ingress
```

```
NAME                         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
                  AGE
cilium-ingress-my-ingress    LoadBalancer   172.16.193.15   <pending>     80:32088/
TCP,443:30332/TCP   25s
```

6. Procedi the section called "Tipo di servizio LoadBalancer" alla configurazione della risorsa Ingress per utilizzare un indirizzo IP allocato da Cilium Load Balancer IPAM the section called "Tipo di servizio NodePort" e/o the section called "Rete host" alla configurazione della risorsa Ingress per utilizzare o ospitare gli indirizzi di rete. NodePort

Tipo di servizio LoadBalancer

Infrastruttura del bilanciatore del carico esistente

Per impostazione predefinita, sia per Cilium Ingress che per Cilium Gateway, Cilium crea uno o più servizi Kubernetes di tipo LoadBalancer per le risorse Ingress/Gateway. Gli attributi dei Servizi creati da Cilium possono essere configurati tramite le risorse dell'ingresso e del gateway. Quando crei risorse ingresso o gateway, l'indirizzo IP o i nomi host esposti esternamente per ingresso o gateway vengono allocati dall'infrastruttura del bilanciatore del carico, che in genere viene fornita da un controller di ingresso o gateway.

Molti controller dell'ingresso e del gateway utilizzano le annotazioni per rilevare e configurare l'infrastruttura del bilanciatore del carico. Le annotazioni per questi controller dell'ingresso e del gateway sono configurate sulle risorse dell'ingresso o del gateway come mostrato negli esempi precedenti. Fai riferimento alla documentazione del controller dell'ingresso o del gateway per le annotazioni che supporta e consulta Kubernetes Ingress documentation e Kubernetes Gateway documentation per un elenco dei controller più diffusi.

> **Important**
>
> Cilium Ingress and Gateway non possono essere utilizzati con Load AWS Balancer Controller AWS e Network Load Balancer NLBs () con EKS Hybrid Nodes. Il tentativo di utilizzarli insieme porta a obiettivi non registrati, poiché l'NLB tenta di connettersi direttamente al Pod

IPs che supporta il tipo di servizio LoadBalancer quando l'NLB `target-type` è impostato su `ip` (requisito per l'utilizzo di NLB con carichi di lavoro in esecuzione su nodi ibridi EKS).

Nessuna infrastruttura del bilanciatore del carico

Se non si dispone di un'infrastruttura di bilanciamento del carico e del corrispondente controller Ingress/Gateway nel proprio ambiente, è LoadBalancer possibile configurare le risorse Ingress/Gateway e i servizi di tipo corrispondente per utilizzare gli indirizzi IP allocati dalla gestione degli indirizzi IP Load Balancer (LB IPAM) di Cilium. Cilium LB IPAM può essere configurato con intervalli di indirizzi IP noti dall'ambiente locale e può utilizzare il supporto Border Gateway Protocol (BGP) integrato di Cilium o gli annunci L2 per pubblicizzare gli indirizzi LoadBalancer IP sulla rete locale.

L'esempio seguente mostra come configurare l'IPAM LB di Cilium con un indirizzo IP da utilizzare per le risorse Ingress/Gateway e come configurare Cilium BGP Control Plane per pubblicizzare l'indirizzo LoadBalancer IP con la rete locale. La funzionalità LB IPAM di Cilium è abilitata per impostazione predefinita, ma non viene attivata finché non viene creata una risorsa `CiliumLoadBalancerIPPool`.

Prerequisiti

- Ingresso e gateway di Cilium installati seguendo le istruzioni in the section called "Installazione dell'ingresso di Cilium" o the section called "Installazione del gateway di Cilium".
- Risorse dell'ingresso o del gateway di Cilium con applicazione di esempio distribuita seguendo le istruzioni in the section called "Implementazione dell'ingresso di Cilium" o the section called "Implementazione del gateway di Cilium".
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non desideri utilizzare BGP, puoi saltare questo prerequisito, ma non sarai in grado di accedere alla tua risorsa Ingress o Gateway finché l'indirizzo LoadBalancer IP assegnato da Cilium LB IPAM non sarà instradabile sulla tua rete locale.

Procedura

1. Facoltativamente, applica una patch alla risorsa Ingress o Gateway per richiedere un indirizzo IP specifico da utilizzare per il tipo di servizio. LoadBalancer Se non richiedi un indirizzo IP specifico, Cilium ne assegnerà uno dall'intervallo di indirizzi IP configurato nella risorsa `CiliumLoadBalancerIPPool` nel passaggio successivo. Nei comandi seguenti, sostituiscilo `LB_IP_ADDRESS` con l'indirizzo IP per richiedere il tipo di servizio. LoadBalancer

Gateway

```
kubectl patch gateway -n default my-gateway --type=merge -p '{
  "spec": {
    "addresses": [{"type": "IPAddress", "value": "LB_IP_ADDRESS"}]
  }
}'
```

Ingresso

```
kubectl patch ingress my-ingress --type=merge -p '{
  "metadata": {"annotations": {"lbipam.cilium.io/ips": "LB_IP_ADDRESS"}}
}'
```

2. Crea un file denominato `cilium-lbip-pool-ingress.yaml` con una risorsa `CiliumLoadBalancerIPPool` per configurare l'intervallo di indirizzi IP del bilanciatore del carico per le tue risorse ingresso/gateway.
   - Se utilizzi l'ingresso di Cilium, Cilium applica automaticamente l'etichetta `cilium.io/ingress: "true"` ai Servizi che crea per le risorse dell'ingresso. Puoi utilizzare questa etichetta nel campo `serviceSelector` della definizione delle risorse `CiliumLoadBalancerIPPool` per selezionare i Servizi idonei per LB IPAM.
   - Se utilizzi il gateway di Cilium, puoi usare l'etichetta `gateway.networking.k8s.io/gateway-name` nei campi `serviceSelector` della definizione delle risorse per selezionare le risorse `CiliumLoadBalancerIPPool` gateway idonee per LB IPAM.
   - Sostituisci `LB_IP_CIDR` con l'intervallo di indirizzi IP da utilizzare per quelli del bilanciatore del carico. Per selezionare un singolo indirizzo IP, utilizza un CIDR `/32`. Per ulteriori informazioni, consulta Gestione degli indirizzi LoadBalancer IP nella documentazione di Cilium.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: bookinfo-pool
spec:
  blocks:
  - cidr: "LB_IP_CIDR"
  serviceSelector:
    # if using Cilium Gateway
    matchExpressions:
```

```
      - { key: gateway.networking.k8s.io/gateway-name, operator: In, values: [ my-
gateway ] }
    # if using Cilium Ingress
    matchLabels:
      cilium.io/ingress: "true"
```

3. Applica la risorsa `CiliumLoadBalancerIPPool` al cluster.

```
kubectl apply -f cilium-lbip-pool-ingress.yaml
```

4. Conferma che un indirizzo IP è stato allocato da Cilium LB IPAM per la risorsa dell'ingresso/gateway.

   Gateway

```
kubectl get gateway my-gateway
```

```
NAME         CLASS    ADDRESS        PROGRAMMED   AGE
my-gateway   cilium   LB_IP_ADDRESS    True         6m41s
```

   Ingresso

```
kubectl get ingress my-ingress
```

```
NAME         CLASS    HOSTS   ADDRESS        PORTS   AGE
my-ingress   cilium   *       LB_IP_ADDRESS   80      10m
```

5. Crea un file denominato `cilium-bgp-advertisement-ingress.yaml` con una `CiliumBGPAdvertisement` risorsa per pubblicizzare l'indirizzo LoadBalancer IP per le risorse Ingress/Gateway. Se non utilizzi un BGP di Cilium, puoi ignorare questo passaggio. L'indirizzo LoadBalancer IP utilizzato per la risorsa Ingress/Gateway deve essere instradabile sulla rete locale per poter interrogare il servizio nel passaggio successivo.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-lb-ip
  labels:
    advertise: bgp
spec:
```

```
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - LoadBalancerIP
      selector:
        # if using Cilium Gateway
        matchExpressions:
          - { key: gateway.networking.k8s.io/gateway-name, operator: In, values:
 [ my-gateway ] }
        # if using Cilium Ingress
        matchLabels:
          cilium.io/ingress: "true"
```

6. Applica la risorsa `CiliumBGPAdvertisement` al cluster.

```
kubectl apply -f cilium-bgp-advertisement-ingress.yaml
```

7. Accedi al servizio utilizzando l'indirizzo IP assegnato da Cilium LB IPAM.

```
curl -s http://LB_IP_ADDRESS:80/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

### Tipo di servizio NodePort

Se non disponi di un'infrastruttura di bilanciamento del carico e del corrispondente controller Ingress nel tuo ambiente, o se gestisci autonomamente l'infrastruttura di bilanciamento del carico o utilizzi il bilanciamento del carico basato su DNS, puoi configurare Cilium Ingress per creare servizi di tipo per le risorse Ingress. NodePort Quando si utilizza NodePort con Cilium Ingress, il tipo di servizio è esposto su una porta su ciascun nodo nell'intervallo di porte NodePort 30000-32767. In questa

modalità, quando il traffico raggiunge qualsiasi nodo del cluster sul NodePort, viene quindi inoltrato a un pod che supporta il servizio, che può trovarsi sullo stesso nodo o su un nodo diverso.

> **Note**
>
> Il supporto per i NodePort servizi di Cilium Gateway è previsto per la versione 1.18.x di Cilium (#27273)

Prerequisiti

- Ingresso di Cilium installato seguendo le istruzioni in the section called "Installazione dell'ingresso di Cilium".
- Risorse dell'ingresso di Cilium con applicazione di esempio distribuita seguendo le istruzioni in the section called "Implementazione dell'ingresso di Cilium".

Procedura

1. Aggiorna la risorsa Ingress esistente `my-ingress` per cambiarla da Tipo di servizio a. LoadBalancer NodePort

   ```
   kubectl patch ingress my-ingress --type=merge -p '{
       "metadata": {"annotations": {"ingress.cilium.io/service-type": "NodePort"}}
   }'
   ```

   Se non hai creato la risorsa dell'ingresso, puoi crearla applicando la seguente definizione dell'ingresso al tuo cluster. Nota, la definizione dell'ingresso riportata di seguito utilizza l'applicazione di esempio Istio Bookinfo descritta in the section called "Implementazione dell'ingresso di Cilium".

   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   metadata:
     name: my-ingress
     namespace: default
     annotations:
       service.beta.kubernetes.io/aws-load-balancer-type: "external"
       "ingress.cilium.io/service-type": "NodePort"
   spec:
   ```

```
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

2. Conferma che il servizio per la risorsa Ingress è stato aggiornato per utilizzare il tipo di servizio. NodePort Nota la porta per il protocollo HTTP nell'output. Nell'esempio seguente, questa è la porta HTTP 32353, che verrà utilizzata in un passaggio successivo per interrogare il servizio. Il vantaggio dell'utilizzo di Cilium Ingress with Service of type NodePort è che è possibile applicare il routing basato su percorsi e host, nonché politiche di rete per il traffico in ingresso, cosa che non è possibile eseguire per un servizio standard di tipo senza Ingress. NodePort

```
kubectl -n default get svc cilium-ingress-my-ingress
```

```
NAME                         TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)
             AGE
cilium-ingress-my-ingress    NodePort   172.16.47.153   <none>        80:32353/
TCP,443:30253/TCP   27m
```

3. Ottieni l'indirizzo IP dei nodi nel cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION               INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

```
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Accedi al tipo di servizio NodePort utilizzando gli indirizzi IP dei tuoi nodi e quelli acquisiti in precedenza. NodePort Nell'esempio seguente l'indirizzo IP del nodo utilizzato è `10.80.146.32` e lo NodePort è`32353`. Sostituisci questi valori per il tuo ambiente.

```
curl -s http://10.80.146.32:32353/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Rete host

Analogamente al tipo di servizio NodePort, se non si dispone di un'infrastruttura di bilanciamento del carico e di un controller Ingress o Gateway, o se si gestisce autonomamente il bilanciamento del carico con un bilanciamento del carico esterno, è possibile configurare Cilium Ingress e Cilium Gateway per esporre le risorse Ingress e Gateway direttamente sulla rete host. Quando la modalità di rete host è abilitata per una risorsa Ingress o Gateway, il tipo LoadBalancer e le NodePort modalità del Servizio vengono automaticamente disabilitati, la modalità di rete host si esclude a vicenda con queste modalità alternative per ogni risorsa Ingress o Gateway. Rispetto alla NodePort modalità Service of type, la modalità di rete host offre una flessibilità aggiuntiva per la gamma di porte che possono essere utilizzate (non è limitata alla NodePort gamma 30000-32767) ed è possibile configurare un sottoinsieme di nodi in cui i proxy Envoy vengono eseguiti sulla rete host.

Prerequisiti

- Ingresso e gateway di Cilium installati seguendo le istruzioni in the section called “Installazione dell’ingresso di Cilium” o the section called “Installazione del gateway di Cilium”.

Procedura

Gateway

1. Crea un archivio denominato `cilium-gateway-host-network.yaml` con i seguenti contenuti.

```
gatewayAPI:
  enabled: true
  hostNetwork:
    enabled: true
    # uncomment to restrict nodes where Envoy proxies run on the host network
    # nodes:
    #   matchLabels:
    #     role: gateway
```

2. Applica la configurazione del gateway di Cilium della rete host al cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  -f cilium-gateway-host-network.yaml
```

Se non hai creato la risorsa del gateway, puoi crearla applicando la seguente definizione di gateway al tuo cluster. La definizione del gateway riportata di seguito utilizza l'applicazione di esempio Istio Bookinfo descritta in the section called "Implementazione del gateway di Cilium". Nell'esempio seguente, la risorsa del gateway è configurata per utilizzare la porta 8111 per il listener HTTP, che è la porta listener condivisa per i proxy Envoy in esecuzione sulla rete host. Se stai utilizzando una porta privilegiata (inferiore a 1023) per la risorsa del gateway, fai riferimento a Cilium documentation per le istruzioni.

```
---
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  listeners:
  - protocol: HTTP
    port: 8111
```

```
    name: web-gw
    allowedRoutes:
      namespaces:
        from: Same
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /details
    backendRefs:
    - name: details
      port: 9080
```

Puoi osservare la configurazione Cilium Envoy applicata con il seguente comando.

```
kubectl get cec cilium-gateway-my-gateway -o yaml
```

Puoi ottenere la porta del listener Envoy per il servizio `cilium-gateway-my-gateway` con il seguente comando. In questo esempio, la porta del listener condiviso è `8111`.

```
kubectl get cec cilium-gateway-my-gateway -o jsonpath={.spec.services[0].ports[0]}
```

3. Ottieni l'indirizzo IP dei nodi nel cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION              INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

```
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Accedi al servizio utilizzando gli indirizzi IP dei nodi e la porta del listener per la risorsa `cilium-gateway-my-gateway`. Nell'esempio seguente l'indirizzo IP del nodo utilizzato è `10.80.146.32` e la porta del listener è 8111. Sostituisci questi valori per il tuo ambiente.

```
curl -s http://10.80.146.32:8111/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

### Ingresso

A causa di un problema con Cilium upstream (#34028), l'ingresso di Cilium in modalità rete host richiede l'utilizzo di `loadbalancerMode: shared`, che crea un unico servizio di tipo ClusterIP per tutte le risorse dell'ingresso nel cluster. Se stai utilizzando una porta privilegiata (inferiore a 1023) per la risorsa dell'ingresso, fai riferimento a Cilium documentation per le istruzioni.

1. Crea un archivio denominato `cilium-ingress-host-network.yaml` con i seguenti contenuti.

```
ingressController:
  enabled: true
  loadbalancerMode: shared
  # This is a workaround for the upstream Cilium issue
  service:
    externalTrafficPolicy: null
```

```
    type: ClusterIP
  hostNetwork:
    enabled: true
    # ensure the port does not conflict with other services on the node
    sharedListenerPort: 8111
    # uncomment to restrict nodes where Envoy proxies run on the host network
    # nodes:
    #   matchLabels:
    #     role: ingress
```

2. Applica la configurazione dell'ingresso di Cilium della rete host al tuo cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  -f cilium-ingress-host-network.yaml
```

Se non hai creato la risorsa dell'ingresso, puoi crearla applicando la seguente definizione dell'ingresso al tuo cluster. La definizione dell'ingresso riportata di seguito utilizza l'applicazione di esempio Istio Bookinfo descritta in the section called "Implementazione dell'ingresso di Cilium".

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
spec:
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

Puoi osservare la configurazione Cilium Envoy applicata con il seguente comando.

```
kubectl get cec -n kube-system cilium-ingress -o yaml
```

Puoi ottenere la porta del listener Envoy per il servizio `cilium-ingress` con il seguente comando. In questo esempio, la porta del listener condiviso è `8111`.

```
kubectl get cec -n kube-system cilium-ingress -o
 jsonpath={.spec.services[0].ports[0]}
```

3. Ottieni l'indirizzo IP dei nodi nel cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION              INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Accedi al servizio utilizzando gli indirizzi IP dei tuoi nodi e la `sharedListenerPort` per la risorsa`cilium-ingress`. Nell'esempio seguente l'indirizzo IP del nodo utilizzato è `10.80.146.32` e la porta del listener è `8111`. Sostituisci questi valori per il tuo ambiente.

```
curl -s http://10.80.146.32:8111/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
```

```
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

## Configurazione dei servizi di tipo LoadBalancer per nodi ibridi

Questo argomento descrive come configurare il bilanciamento del carico di livello 4 (L4) per le applicazioni in esecuzione su Amazon EKS Hybrid Nodes. I servizi Kubernetes di tipo LoadBalancer vengono utilizzati per esporre le applicazioni Kubernetes esterne al cluster. I servizi di tipo LoadBalancer vengono comunemente utilizzati con un'infrastruttura di bilanciamento del carico fisico in ambiente cloud oppure on-premises per servire il traffico del carico di lavoro. Questa infrastruttura di bilanciamento del carico viene in genere fornita con un controller specifico per l'ambiente.

AWS supporta AWS Network Load Balancer (NLB) e Cilium per i servizi di tipo LoadBalancer in esecuzione su nodi ibridi EKS. La decisione di utilizzare NLB o Cilium si basa sulla fonte del traffico dell'applicazione. Se il traffico dell'applicazione proviene da una Regione AWS, AWS consiglia di utilizzare NLB di AWS e il controller del bilanciatore del carico di AWS. Se il traffico delle applicazioni proviene dall'ambiente locale on-premises o edge, AWS consiglia di utilizzare le funzionalità di bilanciamento del carico integrate di Cilium, che possono essere utilizzate con o senza l'infrastruttura di bilanciamento del carico nell'ambiente.

Per il bilanciamento del carico del traffico delle applicazioni di livello 7 (L7), consulta the section called "Configurazione dell'ingresso". Per informazioni generali sul bilanciamento del carico con EKS, consulta Best Practices for Load Balancing.

Network Load Balancer di AWS

Puoi utilizzare il Controller di bilanciatore del carico di AWS e NLB con il tipo di destinazione `ip` per i carichi di lavoro in esecuzione su nodi ibridi. Quando utilizzi il tipo di destinazione `ip`, NLB inoltra il traffico direttamente ai pod, aggirando il percorso di rete del livello di servizio. Affinché NLB raggiunga le destinazioni IP dei pod sui nodi ibridi, i pod CIDR on-premises devono essere instradabili sulla rete on-premises. Inoltre, il Controller del bilanciatore del carico di AWS utilizza webhook e richiede una comunicazione diretta dal piano di controllo EKS. Per ulteriori informazioni, consulta the section called "Configurazione di webhook".

- Consulta the section called "Bilanciamento del carico di rete" per i requisiti di configurazione delle sottoreti, consulta the section called "Installa con Helm" e Best Practices for Load Balancing per ulteriori informazioni sul Network Load Balancer di AWS e il Controller del bilanciatore del carico di AWS.

- Consulta le AWS Load Balancer Controller NLB configurations per le configurazioni che possono essere applicate ai servizi di tipo `LoadBalancer` con il Network Load Balancer di AWS.

Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non desideri utilizzare BGP, devi utilizzare un metodo alternativo per rendere i pod CIDR on-premises instradabili sulla rete on-premises; consulta the section called "CIDR dei pod remoti instradabili" per ulteriori informazioni.
- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.
- eksctl è installato nell'ambiente a riga di comando, consulta Setup eksctl instructions.

Procedura

1. Scaricare una policy IAM per il Controller del bilanciatore del carico AWS che consente di effettuare chiamate alle API AWS per conto dell'utente.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Sostituisci i valori per il nome del cluster (`CLUSTER_NAME`), la Regione AWS (`AWS_REGION`) e l'ID dell'account AWS (`AWS_ACCOUNT_ID`) con le tue impostazioni ed esegui il comando seguente.

```
eksctl create iamserviceaccount \
    --cluster=CLUSTER_NAME \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws:iam::AWS_ACCOUNT_ID:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region AWS_REGION \
```

```
    --approve
```

4. Aggiungi il repository di grafici Helm eks-charts. AWS conserva questo repository su GitHub.

```
helm repo add eks https://aws.github.io/eks-charts
```

5. Aggiorna il repository locale per assicurarti di avere i grafici più recenti.

```
helm repo update eks
```

6. Installazione del Controller del bilanciatore del carico AWS. Sostituisci i valori per il nome del cluster (`CLUSTER_NAME`), la Regione AWS (`AWS_REGION`), l'ID del VPC (`VPC_ID`) e la versione del Controller del bilanciatore del carico di AWS del grafico Helm (`AWS_LBC_HELM_VERSION`) con le tue impostazioni. Puoi trovare la versione più recente del grafico Helm eseguendo `helm search repo eks/aws-load-balancer-controller --versions`. Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi nel Cloud di AWS, puoi eseguire il Controller del bilanciatore del carico di AWS sui nodi cloud seguendo le istruzioni riportate in the section called "AWS Controller del sistema di bilanciamento del carico".

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --version AWS_LBC_HELM_VERSION \
  --set clusterName=CLUSTER_NAME \
  --set region=AWS_REGION \
  --set vpcId=VPC_ID \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

7. Verifica che il Controller del bilanciatore del carico di AWS sia stato installato correttamente.

```
kubectl get -n kube-system deployment aws-load-balancer-controller
```

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

8. Definisci un'applicazione di esempio in un file denominato `tcp-sample-app.yaml`. L'esempio seguente utilizza una semplice implementazione NGINX con una porta TCP.

```
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: tcp-sample-app
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

9. Applica l'implementazione al cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

10 Definisci un servizio di tipo LoadBalancer per l'implementazione in un file denominato `tcp-sample-service.yaml`.

```
apiVersion: v1
kind: Service
metadata:
  name: tcp-sample-service
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
```

```
    app: nginx
```

11 Applica la mappa di configurazione al cluster.

```
kubectl apply -f tcp-sample-service.yaml
```

12 Il provisioning dell'NLB per il Servizio può richiedere alcuni minuti. Una volta effettuato il provisioning dell'NLB, al Servizio verrà assegnato un indirizzo che corrisponde al nome DNS dell'implementazione dell'NLB.

```
kubectl get svc tcp-sample-service
```

```
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP
                                                PORT(S)        AGE
tcp-sample-service   LoadBalancer   172.16.115.212   k8s-default-tcpsampl-xxxxxxxxxx-
xxxxxxxxxxxxxxxx.elb.<region>.amazonaws.com   80:30396/TCP   8s
```

13 Accedi al Servizio utilizzando l'indirizzo dell'NLB.

```
curl k8s-default-tcpsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.<region>.amazonaws.com
```

Di seguito viene riportato un output di esempio.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14 Eliminare tutte le risorse create.

```
kubectl delete -f tcp-sample-service.yaml
kubectl delete -f tcp-sample-app.yaml
```

### Bilanciamento del carico Cilium all'interno del cluster

Cilium può essere utilizzato come bilanciatore del carico all'interno del cluster per i carichi di lavoro in esecuzione su nodi ibridi EKS, il che può essere utile per ambienti che non dispongono di un'infrastruttura di bilanciamento del carico. Le funzionalità di bilanciamento del carico di Cilium

si basano su una combinazione di funzionalità di Cilium, tra cui la sostituzione del kube-proxy, la gestione indirizzi IP (IPAM) del bilanciatore del carico e il piano di controllo BGP. Le responsabilità di queste funzionalità sono descritte di seguito:

- Sostituzione del Cilium proxy-kube: gestisce il traffico del servizio di routing verso i pod di backend.
- IPAM del bilanciatore del carico Cilium: gestisce gli indirizzi IP che possono essere assegnati a servizi di tipo `LoadBalancer`.
- Piano di controllo BGP Cilium: pubblicizza gli indirizzi IP assegnati dalla IPAM del bilanciatore del carico alla rete on-premises.

Se non utilizzi la sostituzione del kube-proxy Cilium, puoi comunque utilizzare l'IPAM del bilanciatore del carico e il Piano di controllo BGP Cilium per allocare e assegnare indirizzi IP per i servizi di tipo LoadBalancer. Se non utilizzi la sostituzione del kube-proxy Cilium, il bilanciamento del carico dei servizi sui pod di backend viene gestito per impostazione predefinita dalle regole kube-proxy e iptables in EKS.

Prerequisiti

- Cilium è stato installato seguendo le istruzioni contenute in the section called "Configurazione della CNI" con o senza la sostituzione di kube-proxy abilitata. La sostituzione del kube-proxy Cilium richiede l'esecuzione di un sistema operativo con un kernel Linux recente come v4.19.57, v5.1.16 o v5.2.0. Tutte le versioni recenti dei sistemi operativi supportati per l'uso con nodi ibridi soddisfano questi criteri, ad eccezione di Red Hat Enterprise Linux (RHEL) 8.x.
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non desideri utilizzare BGP, devi utilizzare un metodo alternativo per rendere i pod CIDR on-premises instradabili sulla rete on-premises; consulta the section called "CIDR dei pod remoti instradabili" per ulteriori informazioni.
- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.

Procedura

1. Crea un file denominato `cilium-lbip-pool-loadbalancer.yaml` con una risorsa `CiliumLoadBalancerIPPool` per configurare l'intervallo di indirizzi IP del bilanciatore del carico per i tuoi servizi di tipo LoadBalancer.

- Sostituisci `LB_IP_CIDR` con l'intervallo di indirizzi IP da utilizzare per quelli del bilanciatore del carico. Per selezionare un singolo indirizzo IP, utilizza un CIDR `/32`. Per ulteriori informazioni, consulta LoadBalancer IP Address Management nella documentazione di Cilium.
- Il campo `serviceSelector` è configurato in modo che corrisponda al nome del servizio che creerai in un passaggio successivo. Con questa configurazione, gli IP di questo pool verranno assegnati solo ai servizi con lo stesso nome `tcp-sample-service`.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: tcp-service-pool
spec:
  blocks:
  - cidr: "LB_IP_CIDR"
  serviceSelector:
    matchLabels:
      io.kubernetes.service.name: tcp-sample-service
```

2. Applica la risorsa `CiliumLoadBalancerIPPool` al cluster.

```
kubectl apply -f cilium-lbip-pool-loadbalancer.yaml
```

3. Verifica che nel pool sia disponibile almeno un indirizzo IP.

```
kubectl get ciliumloadbalancerippools.cilium.io
```

```
NAME               DISABLED   CONFLICTING   IPS AVAILABLE   AGE
tcp-service-pool   false      False         1               24m
```

4. Crea un file denominato `cilium-bgp-advertisement-loadbalancer.yaml` con una risorsa `CiliumBGPAdvertisement` per pubblicizzare l'indirizzo IP del bilanciatore del carico per il servizio che creerai nel passaggio successivo. Se non utilizzi un BGP di Cilium, puoi ignorare questo passaggio. L'indirizzo IP del bilanciatore del carico utilizzato per il servizio deve essere instradabile sulla rete on-premises per consentire all'utente di interrogare il servizio nella Passaggio finale.
   - Il campo `advertisementType` è impostato su `Service` e `service.addresses` è impostato su `LoadBalancerIP` per pubblicizzare solo l'`LoadBalancerIP` per i servizi di tipo `LoadBalancer`.

- Il campo `selector` è configurato in modo che corrisponda al nome del servizio che creerai in un passaggio successivo. Con questa configurazione, verrà pubblicizzato solo l'`LoadBalancerIP` per i servizi con lo stesso nome `tcp-sample-service`.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-tcp-service
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - LoadBalancerIP
      selector:
        matchLabels:
          io.kubernetes.service.name: tcp-sample-service
```

5. Applica la risorsa `CiliumBGPAdvertisement` al cluster. Se non utilizzi un BGP di Cilium, puoi ignorare questo passaggio.

```
kubectl apply -f cilium-bgp-advertisement-loadbalancer.yaml
```

6. Definisci un'applicazione di esempio in un file denominato `tcp-sample-app.yaml`. L'esempio seguente utilizza una semplice implementazione NGINX con una porta TCP.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tcp-sample-app
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
```

```
  spec:
    containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
          - name: tcp
            containerPort: 80
```

7. Applica l'implementazione al cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

8. Definisci un servizio di tipo LoadBalancer per l'implementazione in un file denominato `tcp-sample-service.yaml`.
   - Puoi richiedere un indirizzo IP specifico dal pool di IP del bilanciatore del carico con l'annotazione `lbipam.cilium.io/ips` sull'oggetto del servizio. Puoi rimuovere questa annotazione se non desideri richiedere un indirizzo IP specifico per il servizio.
   - Il campo della specifica `loadBalancerClass` è obbligatorio per impedire al provider cloud di AWS precedente di creare un Classic Load Balancer per il servizio. Nell'esempio seguente, questo è configurato su `io.cilium/bgp-control-plane` per utilizzare il piano di controllo BGP di Cilium come classe di bilanciatore del carico. In alternativa, questo campo può essere configurato su `io.cilium/l2-announcer` per utilizzare la Funzionalità L2 Announcements di Cilium (attualmente in versione beta e non supportata ufficialmente da AWS).

```
apiVersion: v1
kind: Service
metadata:
  name: tcp-sample-service
  namespace: default
  annotations:
    lbipam.cilium.io/ips: "LB_IP_ADDRESS"
spec:
  loadBalancerClass: io.cilium/bgp-control-plane
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

9. Applica il servizio al cluster. Il servizio verrà creato con un indirizzo IP esterno che puoi utilizzare per accedere all'applicazione.

```
kubectl apply -f tcp-sample-service.yaml
```

10. Verifica che il servizio sia stato creato correttamente e che gli sia stato assegnato un IP dal `CiliumLoadBalancerIPPool` creato nel passaggio precedente.

```
kubectl get svc tcp-sample-service
```

```
NAME                 TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)
 AGE
tcp-sample-service   LoadBalancer   172.16.117.76   LB_IP_ADDRESS   80:31129/TCP
 14m
```

11. Se utilizzi Cilium in modalità kube-proxy sostitutivo, puoi confermare che Cilium sta gestendo il bilanciamento del carico per il servizio eseguendo il comando seguente. Nell'output seguente, gli indirizzi `10.86.2.x` sono gli indirizzi IP dei pod di backend per il servizio.

```
kubectl -n kube-system exec ds/cilium -- cilium-dbg service list
```

```
ID   Frontend              Service Type   Backend
...
41   LB_IP_ADDRESS:80/TCP   LoadBalancer   1 => 10.86.2.76:80/TCP (active)
                                           2 => 10.86.2.130:80/TCP (active)
                                           3 => 10.86.2.141:80/TCP (active)
```

12. Conferma che Cilium stia pubblicizzando l'indirizzo IP sulla rete on-premises tramite BGP. Nell'esempio seguente ci sono cinque nodi ibridi, ciascuno dei quali pubblicizza `LB_IP_ADDRESS` per il servizio `tcp-sample-service` sulla rete on-premises.

```
Node                    VRouter      Prefix             NextHop   Age     Attrs
mi-026d6a261e355fba7    NODES_ASN
                  LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-082f73826a163626e    NODES_ASN
                  LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-09183e8a3d755abf6    NODES_ASN
```

```
                  LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-0d78d815980ed202d   NODES_ASN
                  LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-0daa253999fe92daa   NODES_ASN
                  LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
```

13 Accedi al servizio utilizzando l'indirizzo IP del bilanciatore del carico assegnato.

```
curl LB_IP_ADDRESS
```

Di seguito viene riportato un output di esempio.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14 Eliminare tutte le risorse create.

```
kubectl delete -f tcp-sample-service.yaml
kubectl delete -f tcp-sample-app.yaml
kubectl delete -f cilium-lb-ip-pool.yaml
kubectl delete -f cilium-bgp-advertisement.yaml
```

## Configurazione delle policy di rete Kubernetes per i nodi ibridi

AWS supporta le politiche di rete Kubernetes (Layer 3/Layer 4) per il traffico in ingresso e in uscita dei pod quando si utilizza Cilium come CNI con EKS Hybrid Nodes. Se utilizzi cluster EKS con nodi nel AWS cloud, AWS supporta Amazon VPC CNI for Kubernetes Network Policies.

Questo argomento descrive come configurare le policy di rete Cilium e Kubernetes con EKS Hybrid Nodes. Per informazioni dettagliate sulle policy di rete Kubernetes, consulta le Policy di rete Kubernetes nella documentazione di Kubernetes.

Configurazione delle policy di rete

Considerazioni

- AWS supporta le politiche di rete Kubernetes upstream e le specifiche per l'ingresso e l'uscita dei pod. AWS attualmente non supporta o. `CiliumNetworkPolicy` `CiliumClusterwideNetworkPolicy`
- Il valore Helm `policyEnforcementMode` può essere utilizzato per controllare il comportamento predefinito di applicazione delle policy di Cilium. Il comportamento predefinito consente tutto il traffico in uscita e in ingresso. Quando un endpoint viene selezionato da una policy di rete, passa allo stato predefinito di negazione, in cui è consentito solo il traffico esplicitamente consentito. Consulta la documentazione di Cilium per ulteriori informazioni su default policy mode e policy enforcement modes.
- Se stai passando `policyEnforcementMode` a un'installazione Cilium esistente, devi riavviare l'agente Cilium DaemonSet per applicare la nuova modalità di applicazione delle policy.
- Usa `namespaceSelector` e `podSelector` per consentire o negare il traffico di to/from namespace e pod con etichette corrispondenti. `namespaceSelector` e `podSelector` possono essere usati con `matchLabels` o `matchExpressions` per selezionare namespace e pod in base alle relative etichette.
- Utilizzare `ingress.ports` e consentire o `egress.ports` negare porte e protocolli di traffico. to/from
- Il `ipBlock` campo non può essere utilizzato per consentire o negare selettivamente gli indirizzi IP del Traffic to/from Pod (#9209). L'uso `ipBlock` dei selettori per gli IP dei nodi è una funzionalità beta di Cilium e non è supportato da. AWS
- Consulta la NetworkPolicy risorsa nella documentazione di Kubernetes per informazioni sui campi disponibili per le politiche di rete di Kubernetes.

Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".
- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.

Procedura

La procedura seguente imposta le policy di rete per un'applicazione di microservizi di esempio, in modo che i componenti possano comunicare solo con altri componenti necessari per il

funzionamento dell'applicazione. La procedura utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

L'applicazione Bookinfo è composta da quattro microservizi separati con le seguenti relazioni:

- productpage. Il microservizio productpage chiama i dettagli ed esamina i microservizi per popolare la pagina.
- details. Il microservizio details contiene informazioni sui libri.
- reviews. Il microservizio reviews contiene le recensioni dei libri. Chiama anche il microservizio di rating.
- ratings. Il microservizio ratings contiene le informazioni sulla classificazione dei libri che accompagnano la recensione del libro.

1. Crea un'applicazione di esempio.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

2. Conferma che l'applicazione funzioni correttamente e annota l'indirizzo IP del pod per il microservizio productpage. Utilizzerai questo indirizzo IP del pod per interrogare ogni microservizio nei passaggi successivi.

```
kubectl get pods -o wide
```

```
NAME                              READY   STATUS    RESTARTS   AGE   IP
 NODE
details-v1-766844796b-9wff2       1/1     Running   0          7s    10.86.3.7
 mi-0daa253999fe92daa
productpage-v1-54bb874995-lwfgg   1/1     Running   0          7s    10.86.2.193
 mi-082f73826a163626e
ratings-v1-5dc79b6bcd-59njm       1/1     Running   0          7s    10.86.2.232
 mi-082f73826a163626e
reviews-v1-598b896c9d-p2289       1/1     Running   0          7s    10.86.2.47
 mi-026d6a261e355fba7
reviews-v2-556d6457d-djktc        1/1     Running   0          7s    10.86.3.58
 mi-0daa253999fe92daa
reviews-v3-564544b4d6-g8hh4       1/1     Running   0          7s    10.86.2.69
 mi-09183e8a3d755abf6
```

3. Crea un pod che verrà utilizzato ovunque per testare le policy di rete. Ricorda che il pod viene creato nel namespace `default` con l'etichetta `access: true`.

```
kubectl run curl-pod --image=curlimages/curl -i --tty --labels=access=true --
namespace=default --overrides='{"spec": { "nodeSelector": {"eks.amazonaws.com/
compute-type": "hybrid"}}}' -- /bin/sh
```

4. Testa l'accesso al microservizio productpage. Nell'esempio seguente, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo ambiente.

```
curl -s http://10.86.2.193:9080/productpage | grep -o "<title>.*</title>"
```

```
<title>Simple Bookstore App</title>
```

5. Puoi uscire dal pod test curl digitando `exit` e puoi ricollegarti al pod eseguendo il seguente comando.

```
kubectl attach curl-pod -c curl-pod -i -t
```

6. Per dimostrare gli effetti delle politiche di rete nei passaggi seguenti, creiamo innanzitutto una politica di rete che neghi tutto il traffico per i microservizi. BookInfo Crea un file chiamato `network-policy-deny-bookinfo.yaml` che definisce la policy di negazione della rete.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-bookinfo
  namespace: default
spec:
  podSelector:
    matchExpressions:
    - key: app
      operator: In
      values: ["productpage", "details", "reviews", "ratings"]
  policyTypes:
  - Ingress
  - Egress
```

7. Applica la policy di negazione della rete al cluster.

```
kubectl apply -f network-policy-default-deny-bookinfo.yaml
```

8. Verifica l'accesso all'applicazione. BookInfo Nell'esempio seguente, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo ambiente.

```
curl http://10.86.2.193:9080/productpage --max-time 10
```

```
curl: (28) Connection timed out after 10001 milliseconds
```

9. Crea un file chiamato `network-policy-productpage.yaml` che definisce la policy di rete di productpage. La policy ha le seguenti regole:
   - Consente il traffico in ingresso dai pod con l'etichetta `access: true` (il pod curl creato nel passaggio precedente)
   - Consente il traffico TCP in uscita sulla porta `9080` per i microservizi details, reviews e ratings
   - consente il TCP/UDP traffico in uscita sulla porta `53` per CoredNS che viene eseguito nello spazio dei nomi `kube-system`

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: productpage-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: productpage
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
  egress:
  - to:
    - podSelector:
        matchExpressions:
```

```
        - key: app
          operator: In
          values: ["details", "reviews", "ratings"]
    ports:
    - port: 9080
      protocol: TCP
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
```

10. Applica la policy di rete productpage al cluster.

```
kubectl apply -f network-policy-productpage.yaml
```

11. Connettiti al pod curl e testa l'accesso all'applicazione Bookinfo. Ora l'accesso al microservizio productpage è consentito, ma gli altri microservizi vengono comunque negati perché sono ancora soggetti alla policy di negazione della rete. Negli esempi seguenti, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo ambiente.

```
kubectl attach curl-pod -c curl-pod -i -t
```

```
curl -s http://10.86.2.193:9080/productpage | grep -o "<title>.*</title>"
<title>Simple Bookstore App</title>
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1
{"error": "Sorry, product details are currently unavailable for this book."}
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1/reviews
{"error": "Sorry, product reviews are currently unavailable for this book."}
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1/ratings
{"error": "Sorry, product ratings are currently unavailable for this book."}
```

12. Crea un file chiamato `network-policy-details.yaml` che definisce la policy di rete di details. La policy consente solo il traffico in ingresso dal microservizio productpage.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: details-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: details
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: productpage
```

13. Crea un file chiamato `network-policy-reviews.yaml` che definisce la policy di rete di reviews. La policy consente solo il traffico in ingresso dal microservizio productpage e solo il traffico in uscita verso il microservizio ratings e CoreDNS.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: reviews-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: reviews
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
```

```
        matchLabels:
          app: productpage
  egress:
  - to:
    - podSelector:
        matchLabels:
          app: ratings
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
```

14. Crea un file chiamato `network-policy-ratings.yaml` che definisce la policy della rete di ratings. La policy consente solo il traffico in ingresso dai microservizi productpage e reviews.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ratings-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: ratings
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchExpressions:
        - key: app
          operator: In
          values: ["productpage", "reviews"]
```

15. Applica le policy di rete per details, reviews e ratings al cluster.

```
kubectl apply -f network-policy-details.yaml
kubectl apply -f network-policy-reviews.yaml
kubectl apply -f network-policy-ratings.yaml
```

16. Connettiti al pod curl e testa l'accesso all'applicazione Bookinfo. Negli esempi seguenti, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo ambiente.

```
kubectl attach curl-pod -c curl-pod -i -t
```

Testa il microservice details.

```
curl -s http://10.86.2.193:9080/api/v1/products/1
```

```
{"id": 1, "author": "William Shakespeare", "year": 1595, "type": "paperback",
 "pages": 200, "publisher": "PublisherA", "language": "English", "ISBN-10":
 "1234567890", "ISBN-13": "123-1234567890"}
```

Testa il microservizio reviews.

```
curl -s http://10.86.2.193:9080/api/v1/products/1/reviews
```

```
{"id": "1", "podname": "reviews-v1-598b896c9d-p2289", "clustername": "null",
 "reviews": [{"reviewer": "Reviewer1", "text": "An extremely entertaining play
 by Shakespeare. The slapstick humour is refreshing!"}, {"reviewer": "Reviewer2",
 "text": "Absolutely fun and entertaining. The play lacks thematic depth when
 compared to other plays by Shakespeare."}]}
```

Testa il microservizio ratings.

```
curl -s http://10.86.2.193:9080/api/v1/products/1/ratings
```

```
{"id": 1, "ratings": {"Reviewer1": 5, "Reviewer2": 4}}
```

17. Elimina tutte le risorse che hai creato con questa procedura.

```
kubectl delete -f network-policy-deny-bookinfo.yaml
```

```
kubectl delete -f network-policy-productpage.yaml
kubectl delete -f network-policy-details.yaml
kubectl delete -f network-policy-reviews.yaml
kubectl delete -f network-policy-ratings.yaml
kubectl delete -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
kubectl delete pod curl-pod
```

# Concetti per nodi ibridi

Con Amazon EKS Hybrid Nodes, unisci macchine fisiche o virtuali in esecuzione in ambienti on-premises o edge a cluster Amazon EKS in esecuzione nel cloud AWS. Questo approccio offre molti vantaggi, ma introduce anche nuovi concetti e architetture di rete per chi ha dimestichezza con l'esecuzione di cluster Kubernetes in un unico ambiente di rete.

Le sezioni seguenti approfondiscono Kubernetes e i concetti di rete per i nodi ibridi EKS e descrivono in dettaglio come il traffico fluisce attraverso l'architettura ibrida. Queste sezioni richiedono che tu abbia una conoscenza di base delle reti Kubernetes, come i concetti di pod, nodi, servizi, piano di controllo Kubernetes, kubelet e kube-proxy.

Ti consigliamo di leggere queste pagine in ordine, iniziando con the section called "Concetti di rete", poi con the section called "Concetti di Kubernetes" e infine con the section called "Flussi di traffico".

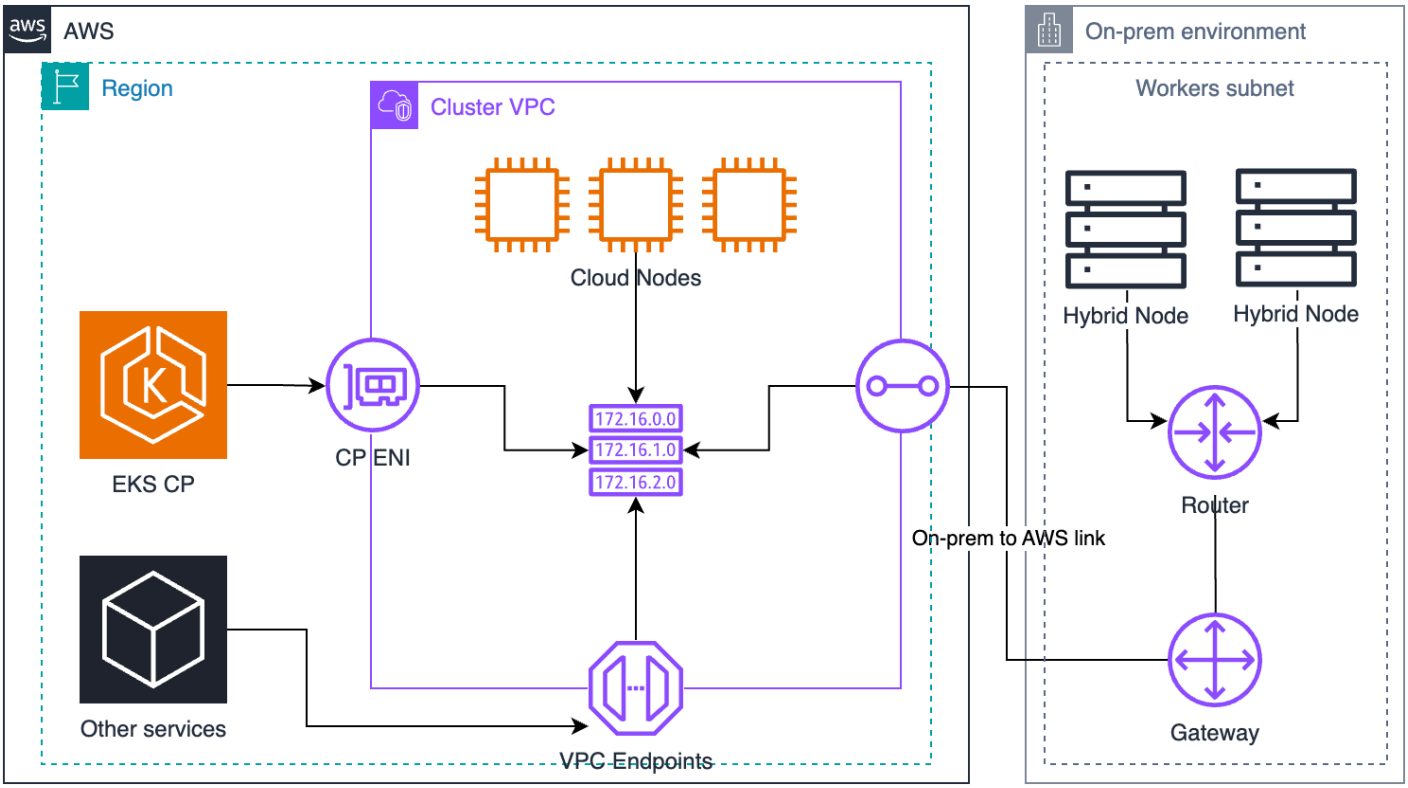
Argomenti

- Concetti di rete per nodi ibridi
- Concetti di Kubernetes per nodi ibridi
- Flussi di traffico di rete per nodi ibridi

## Concetti di rete per nodi ibridi

Questa sezione descrive in dettaglio i concetti di rete principali e i vincoli da considerare durante la progettazione della topologia di rete per EKS Hybrid Nodes.

Concetti di rete per EKS Hybrid Nodes




VPC come hub di rete

Tutto il traffico che supera i limiti del cloud viene instradato attraverso il tuo VPC. Ciò include il traffico tra il piano di controllo EKS o i pod che arrivano AWS ai nodi ibridi o ai pod in esecuzione su di essi. Puoi considerare il VPC del cluster come all'hub di rete tra i nodi ibridi e il resto del cluster. Questa architettura offre pieno controllo del traffico e del relativo instradamento, ma ti impone anche la responsabilità di configurare correttamente instradamenti, gruppi di sicurezza e firewall per il VPC.

Dal piano di controllo (control-plane) EKS al VPC

Il piano di controllo EKS collega Elastic Network Interfaces (ENIs) al tuo VPC. Questi ENIs gestiscono il traffico da e verso il server API EKS. È possibile controllare il posizionamento del piano di controllo EKS ENIs durante la configurazione del cluster, poiché EKS si collega ENIs alle sottoreti passate durante la creazione del cluster.

EKS associa i gruppi di sicurezza a quelli ENIs che EKS collega alle sottoreti. Questi gruppi di sicurezza consentono il traffico da e verso il piano di controllo EKS attraverso il. ENIs Questo è importante per i nodi ibridi EKS perché è necessario consentire il traffico dai nodi ibridi e dai pod in esecuzione su di essi al piano ENIs di controllo EKS.

### Reti di nodi remoti

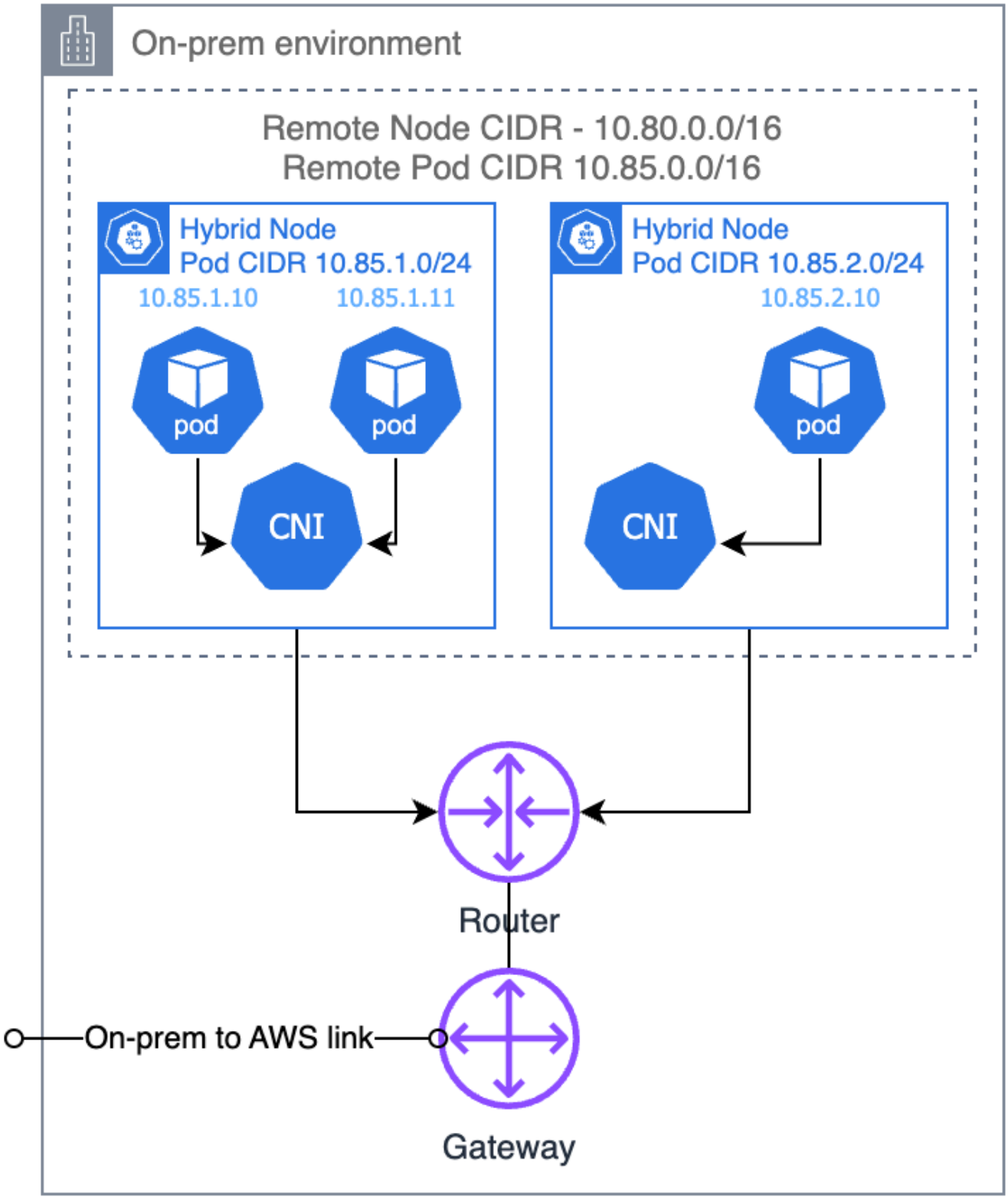
Le reti a nodi remoti, in particolare il nodo remoto CIDRs, sono gli intervalli IPs assegnati alle macchine utilizzate come nodi ibridi. Quando esegui il provisioning dei nodi ibridi, questi risiedono nel data center on-premises o nella posizione edge, che è un dominio di rete diverso dal piano di controllo (control-plane) EKS e dal VPC. Ogni nodo ibrido ha uno o più indirizzi IP da un CIDR del nodo remoto distinto dalle sottoreti del tuo VPC.

Il cluster EKS viene configurato con questi nodi remoti CIDRs in modo che EKS sappia come indirizzare tutto il traffico destinato ai nodi ibridi IPs attraverso il VPC del cluster, ad esempio le richieste all'API kubelet. Le connessioni all'`kubelet`API vengono utilizzate nei comandi`kubectl attach`,, `kubectl cp`, `kubectl exec`, `kubectl logs` e. `kubectl port-forward`

### Reti di pod remoti

Le reti di pod remoti sono gli intervalli IPs assegnati ai pod in esecuzione sui nodi ibridi. In genere, bisogna configurare la Container Network Interface (CNI) con questi intervalli e la funzionalità di Gestione indirizzi IP (IPAM) della CNI si occupa di assegnare una parte di tali intervalli a ciascun nodo ibrido. Quando crei un pod, la CNI assegna un IP al pod dalla sezione allocata al nodo in cui il pod è stato pianificato.

Puoi configurare il cluster EKS con questi pod remoti CIDRs in modo che il piano di controllo EKS sappia instradare tutto il traffico destinato ai pod in esecuzione sui nodi ibridi attraverso il VPC del cluster, ad esempio la comunicazione con i webhook.




Dall'ambiente on-premises al VPC

La rete on-premises utilizzata per i nodi ibridi deve essere indirizzata al VPC usato per il cluster EKS. Sono disponibili diverse opzioni di connettività Network-to-Amazon VPC per connettere la rete locale a un VPC. Inoltre, puoi utilizzare la tua soluzione VPN.

È importante configurare correttamente il routing sul lato AWS cloud nel VPC e nella rete locale, in modo che entrambe le reti instradino il traffico giusto attraverso la connessione per le due reti.

Nel VPC, tutto il traffico diretto alle reti di nodi e pod remoti deve essere instradato attraverso la connessione alla rete on-premises (denominata "gateway"). Se alcune sottoreti hanno tabelle di routing diverse, devi configurare ogni tabella con gli instradamenti per i nodi ibridi e i pod in esecuzione su di essi. Questo vale per le sottoreti a cui è collegato il piano di controllo EKS e per ENIs le sottoreti che contengono EC2 nodi o pod che devono comunicare con i nodi ibridi.

Nella rete locale, è necessario configurare la rete per consentire il traffico da e verso il VPC del cluster EKS e AWS gli altri servizi richiesti per i nodi ibridi. Il traffico per il cluster EKS attraversa il gateway in entrambe le direzioni.

Vincoli di rete

Rete completamente instradata

Il vincolo principale è che il piano di controllo (control-plane) EKS e tutti i nodi, cloud o ibridi, devono formare una rete completamente instradata. Ciò significa che tutti i nodi devono essere in grado di raggiungersi a vicenda al terzo livello, tramite indirizzo IP.

Il piano di controllo (control-plane) EKS e i nodi cloud sono già raggiungibili a vicenda perché si trovano in una rete piatta (il VPC). I nodi ibridi, tuttavia, si trovano in un dominio di rete diverso. Ecco perché è necessario configurare un routing aggiuntivo nel VPC e nella rete on-premises per instradare il traffico tra i nodi ibridi e il resto del cluster. Se i nodi ibridi sono raggiungibili a vicenda e dal VPC, i nodi ibridi possono trovarsi in un'unica rete piatta o in più reti segmentate.

Pod remoto indirizzabile CIDRs

Affinché il piano di controllo (control-plane) EKS comunichi con i pod in esecuzione su nodi ibridi (ad esempio, webhook o Metrics Server) o affinché i pod in esecuzione su nodi cloud comunichino con quelli in esecuzione su nodi ibridi (comunicazione est-ovest del carico di lavoro), il CIDR del pod remoto deve essere instradabile dal VPC. Ciò significa che il VPC deve essere in grado di indirizzare il traffico verso il pod CIDRs attraverso il gateway verso la rete locale e che la rete locale deve essere in grado di indirizzare il traffico di un pod verso il nodo giusto.

È importante notare la distinzione tra i requisiti di routing dei pod nel VPC e on-premises. Il VPC deve solo sapere che tutto il traffico diretto a un pod remoto deve passare attraverso il gateway. Se disponi di un solo CIDR del pod remoto, è necessario un solo instradamento.

Questo requisito è valido per tutti gli hop della rete on-premises fino al router locale nella stessa sottorete dei nodi ibridi. Questo è l'unico router che deve conoscere la sezione CIDR del pod assegnata a ciascun nodo, assicurandosi che il traffico per un pod specifico venga recapitato al nodo in cui è stato pianificato.

Puoi scegliere di propagare queste route per il pod locale CIDRs dal router locale locale alle tabelle di routing VPC, ma non è necessario. Se il tuo pod locale CIDRs cambia frequentemente e le tabelle di routing VPC devono essere aggiornate per rispecchiare la modifica del CIDRs pod, ti consigliamo di propagare il pod locale CIDRs alle tabelle di routing VPC, ma questo non è comune.

Nota, il vincolo per rendere instradabile il pod locale è facoltativo. CIDRs Se non è necessario eseguire webhook sui nodi ibridi o disporre di pod sui nodi cloud, dialoga con i pod sui nodi ibridi, non è necessario configurare il routing per il pod sulla rete locale. CIDRs

Perché il pod locale CIDRs deve essere instradabile con nodi ibridi?

Quando si utilizza EKS con il VPC CNI per i nodi cloud, il VPC CNI esegue l'assegnazione direttamente IPs dal VPC ai pod. Ciò significa che non è necessario alcun routing speciale, poiché sia i cloud pod che il piano di controllo EKS possono raggiungere direttamente il Pod. IPs

Quando vengono eseguiti in locale (e insieme ad altri CNIs nel cloud), i pod vengono in genere eseguiti in una rete overlay isolata e il CNI si occupa della distribuzione del traffico tra i pod. Ciò avviene in genere tramite l'incapsulamento: il CNI converte il pod-to-pod traffico in node-to-node traffico, occupandosi dell'incapsulamento e del deincapsulamento su entrambe le estremità. In questo modo, non è necessaria alcuna configurazione aggiuntiva sui nodi e sui router.

La rete con nodi ibridi è unica perché presenta una combinazione di entrambe le topologie: il piano di controllo (control-plane) EKS e i nodi cloud (con la CNI del VPC) si aspettano una rete piatta che includa nodi e pod, mentre i pod in esecuzione su nodi ibridi si trovano in una rete sovrapposta utilizzando VXLAN per l'incapsulamento (impostazione predefinita in Cilium). I pod in esecuzione su nodi ibridi possono raggiungere il piano di controllo (control-plane) EKS e i pod in esecuzione su nodi cloud, presupponendo che la rete on-premises possa essere instradata verso il VPC. Tuttavia, senza il routing del pod CIDRs sulla rete locale, qualsiasi traffico che ritorna a un IP del pod locale alla fine verrà interrotto se la rete non sa come raggiungere la rete overlay e indirizzarlo ai nodi corretti.

# Concetti di Kubernetes per nodi ibridi

Questa pagina descrive in dettaglio i concetti chiave di Kubernetes alla base dell'architettura di sistema dei nodi ibridi EKS.

Piano di controllo EKS nel VPC

Gli IP del piano di controllo EKS ENI sono memorizzati nell'oggetto `kubernetes` `Endpoints` nel namespace `default`. Quando EKS crea nuovi ENI o rimuove quelli più vecchi, EKS aggiorna questo oggetto in modo che l'elenco degli IP sia sempre aggiornato.

È possibile utilizzare questi endpoint tramite il Servizio `kubernetes`, anche nel namespace `default`. A questo servizio, di tipo `ClusterIP`, viene sempre assegnato il primo IP del servizio CIDR del cluster. Ad esempio, per il servizio CIDR `172.16.0.0/16`, l'IP del servizio sarà `172.16.0.1`.

In genere, questo è il modo in cui i pod (indipendentemente dal fatto che siano in esecuzione nel cloud o nei nodi ibridi) accedono al server API EKS Kubernetes. I pod utilizzano l'IP del servizio come IP di destinazione, che viene tradotto negli IP effettivi di uno degli ENI del piano di controllo EKS. L'eccezione principale è `kube-proxy`, perché imposta la traduzione.

Endpoint del server API EKS

L'IP del servizio `kubernetes` non è l'unico modo per accedere al server API EKS. EKS crea anche un nome DNS Route53 quando si crea il cluster. Questo è il campo `endpoint` del cluster EKS quando si richiama l'azione dell'API EKS `DescribeCluster`.

```
{
    "cluster": {
        "endpoint": "https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com",
        "name": "my-cluster",
        "status": "ACTIVE"
    }
}
```

In un cluster di accesso pubblico agli endpoint o in uno di accesso pubblico e privato agli endpoint, i nodi ibridi risolveranno questo nome DNS in un IP pubblico per impostazione predefinita, instradabile tramite Internet. In un cluster privato di accesso agli endpoint, il nome DNS si risolve negli IP privati del piano di controllo EKS ENI.

Ecco come `kubelet` e `kube-proxy` accedono al server dell'API Kubernetes. Se desideri che tutto il traffico del cluster Kubernetes fluisca attraverso il VPC, devi configurare il cluster in modalità di accesso privato o modificare il server DNS on-premises per risolvere l'endpoint del cluster EKS sugli IP privati del piano di controllo EKS ENI.

Endpoint **`kubelet`**

Il `kubelet` espone diversi endpoint REST, consentendo ad altre parti del sistema di interagire e raccogliere informazioni da ciascun nodo. Nella maggior parte dei cluster, la maggior parte del traffico diretto al server `kubelet` proviene dal piano di controllo, ma anche alcuni agenti di monitoraggio potrebbero interagire con esso.

Tramite questa interfaccia, `kubelet` gestisce varie richieste: recupero dei log (`kubectl logs`), esecuzione di comandi all'interno dei container (`kubectl exec`) e inoltro del traffico (`kubectl port-forward`). Ognuna di queste richieste interagisce con il runtime del container sottostante attraverso il `kubelet`, risultando perfetta per gli amministratori e gli sviluppatori del cluster.

Il consumatore più comune di questa API è il server API Kubernetes. Quando utilizzi uno dei comandi `kubectl` menzionati in precedenza, `kubectl` invia una richiesta API al server API, che quindi chiama l'API `kubelet` del nodo su cui è in esecuzione il pod. Questo è il motivo principale per cui l'IP del nodo deve essere raggiungibile dal piano di controllo EKS e perché, anche se i pod sono in funzione, non sarà possibile accedere ai loro log o `exec` se il percorso del nodo non è configurato correttamente.

IP del nodo

Quando il piano di controllo EKS comunica con un nodo, utilizza uno degli indirizzi riportati nello stato dell'oggetto `Node` (`status.addresses`).

Con i nodi cloud EKS, è normale che il kubelet riporti l'IP privato dell'istanza EC2 come un `InternalIP` durante la registrazione del nodo. Questo IP viene quindi convalidato dal Cloud Controller Manager (CCM) assicurandosi che appartenga all'istanza EC2. Inoltre, il CCM in genere aggiunge gli IP (come `ExternalIP`) pubblici e i nomi DNS (`InternalDNS`e `ExternalDNS`) dell'istanza allo stato del nodo.

Tuttavia, non esiste un CCM per i nodi ibridi. Quando si registra un nodo ibrido con EKS Hybrid Nodes CLI (`nodeadm`), configura il kubelet per riportare l'IP della macchina direttamente nello stato del nodo, senza CCM.

```
apiVersion: v1
```

```
kind: Node
metadata:
  name: my-node-1
spec:
  providerID: eks-hybrid:///us-west-2/my-cluster/my-node-1
status:
  addresses:
  - address: 10.1.1.236
    type: InternalIP
  - address: my-node-1
    type: Hostname
```

Se la macchina ha più IP, il kubelet ne selezionerà uno seguendo la propria logica. Puoi controllare l'IP selezionato con il flag `--node-ip`, che puoi passare in configurazione `nodeadm` in `spec.kubelet.flags`. Solo l'IP riportato nell'oggetto `Node` necessita di un percorso dal VPC. Le tue macchine possono avere altri IP che non sono raggiungibili dal cloud.

### kube-proxy

`kube-proxy` è responsabile dell'implementazione dell'astrazione del servizio a livello di rete di ciascun nodo. Funziona come proxy di rete e bilanciatore del carico per il traffico destinato ai servizi Kubernetes. Monitorando continuamente il server API Kubernetes per rilevare eventuali modifiche relative ai servizi e agli endpoint, `kube-proxy` aggiorna dinamicamente le regole di rete dell'host sottostante per garantire che il traffico venga indirizzato correttamente.

In modalità `iptables`, `kube-proxy` programma diverse catene `netfilter` per gestire il traffico di servizio. Le regole formano la seguente gerarchia:

1. Catena KUBE-SERVICES: il punto di ingresso per tutto il traffico di servizio. Ha regole che corrispondono a ogni `ClusterIP` e porta del servizio.
2. Catene KUBE-SVC-XXX: le catene specifiche dei servizi hanno regole di bilanciamento del carico per ogni servizio.
3. Catene KUBE-SEP-XXX: le catene specifiche per endpoint hanno le regole effettive. `DNAT`

Esaminiamo cosa succede per un servizio `test-server` nel namespace `default`: * Service ClusterIP: `172.16.31.14` * Service port: `80` * Backing pods: `10.2.0.110`, `10.2.1.39`, e `10.2.2.254`

Quando controlliamo le regole `iptables` (usando `iptables-save 0 grep -A10 KUBE-SERVICES`):

1. Nella catena KUBE-SERVICES, troviamo una regola corrispondente al servizio:

```
-A KUBE-SERVICES -d 172.16.31.14/32 -p tcp -m comment --comment "default/test-server
 cluster IP" -m tcp --dport 80 -j KUBE-SVC-XYZABC123456
```

   - Questa regola corrisponde ai pacchetti destinati a 172.16.31.14:80
   - Il commento indica a cosa serve questa regola: `default/test-server cluster IP`
   - I pacchetti corrispondenti saltano alla catena `KUBE-SVC-XYZABC123456`
2. La catena KUBE-SVC-XYZABC123456 ha regole di bilanciamento del carico basate sulla probabilità:

```
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.33333333349 -j
 KUBE-SEP-POD1XYZABC
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.50000000000 -j
 KUBE-SEP-POD2XYZABC
-A KUBE-SVC-XYZABC123456 -j KUBE-SEP-POD3XYZABC
```

   - Prima regola: 33,3% di probabilità di saltare a `KUBE-SEP-POD1XYZABC`
   - Seconda regola: 50% di probabilità che il traffico rimanente (33,3% del totale) passi a `KUBE-SEP-POD2XYZABC`
   - Ultima regola: tutto il traffico rimanente (33,3% del totale) passa a `KUBE-SEP-POD3XYZABC`
3. Le singole catene KUBE-SEP-XXX eseguono il DNAT (Destination NAT):

```
-A KUBE-SEP-POD1XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.0.110:80
-A KUBE-SEP-POD2XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.1.39:80
-A KUBE-SEP-POD3XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.2.254:80
```

   - Queste regole DNAT riscrivono l'IP e la porta di destinazione per indirizzare il traffico verso pod specifici.
   - Ogni regola gestisce circa il 33,3% del traffico, garantendo un bilanciamento uniforme del carico tra `10.2.0.110`, `10.2.1.39` e `10.2.2.254`.

Questa struttura a catena a più livelli consente a `kube-proxy` di implementare in modo efficiente il bilanciamento e il reindirizzamento del carico di servizio mediante la manipolazione dei pacchetti a livello di kernel, senza richiedere un processo proxy nel percorso dei dati.

Impatto sulle operazioni di Kubernetes

Un'interruzione `kube-proxy` su un nodo impedisce a quel nodo di indirizzare correttamente il traffico del Servizio, causando timeout o connessioni non riuscite per i pod che si basano sui servizi del cluster. Ciò può essere particolarmente problematico quando un nodo viene registrato per la prima volta. Il CNI deve comunicare con il server dell'API Kubernetes per ottenere informazioni, come il pod CIDR del nodo, prima di poter configurare qualsiasi rete di pod. A tale scopo, utilizza l'IP del servizio `kubernetes`. Tuttavia, se `kube-proxy` non è riuscito ad avviarlo o a impostare le regole `iptables` corrette, le richieste inviate all'IP del servizio `kubernetes` non vengono tradotte negli IP effettivi degli ENI del piano di controllo EKS. Di conseguenza, il CNI entrerà in un ciclo di arresto e nessuno dei pod sarà in grado di funzionare correttamente.

Sappiamo che i pod utilizzano l'IP del servizio `kubernetes` per comunicare con il server dell'API Kubernetes, ma `kube-proxy` deve prima impostare le regole `iptables` per farlo funzionare.

Come `kube-proxy` comunica con il server API?

`kube-proxy` deve essere configurato per utilizzare gli IP/s effettivi del server dell'API Kubernetes o un nome DNS che li risolva. Nel caso di EKS, EKS configura il predefinito `kube-proxy` in modo che punti al nome DNS Route53 creato da EKS quando si crea il cluster. Puoi vedere questo valore nella ConfigMap `kube-proxy` nel namespace `kube-system`. Il contenuto di questa ConfigMap è un `kubeconfig` che viene iniettato nel pod `kube-proxy`, quindi cerca il campo `clusters0.cluster.server`. Questo valore corrisponderà al campo `endpoint` del cluster EKS (quando chiama l'API `DescribeCluster` di EKS).

```
apiVersion: v1
data:
  kubeconfig: |-
    kind: Config
    apiVersion: v1
    clusters:
    - cluster:
        certificate-authority: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        server: https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com
      name: default
    contexts:
    - context:
        cluster: default
        namespace: default
        user: default
```

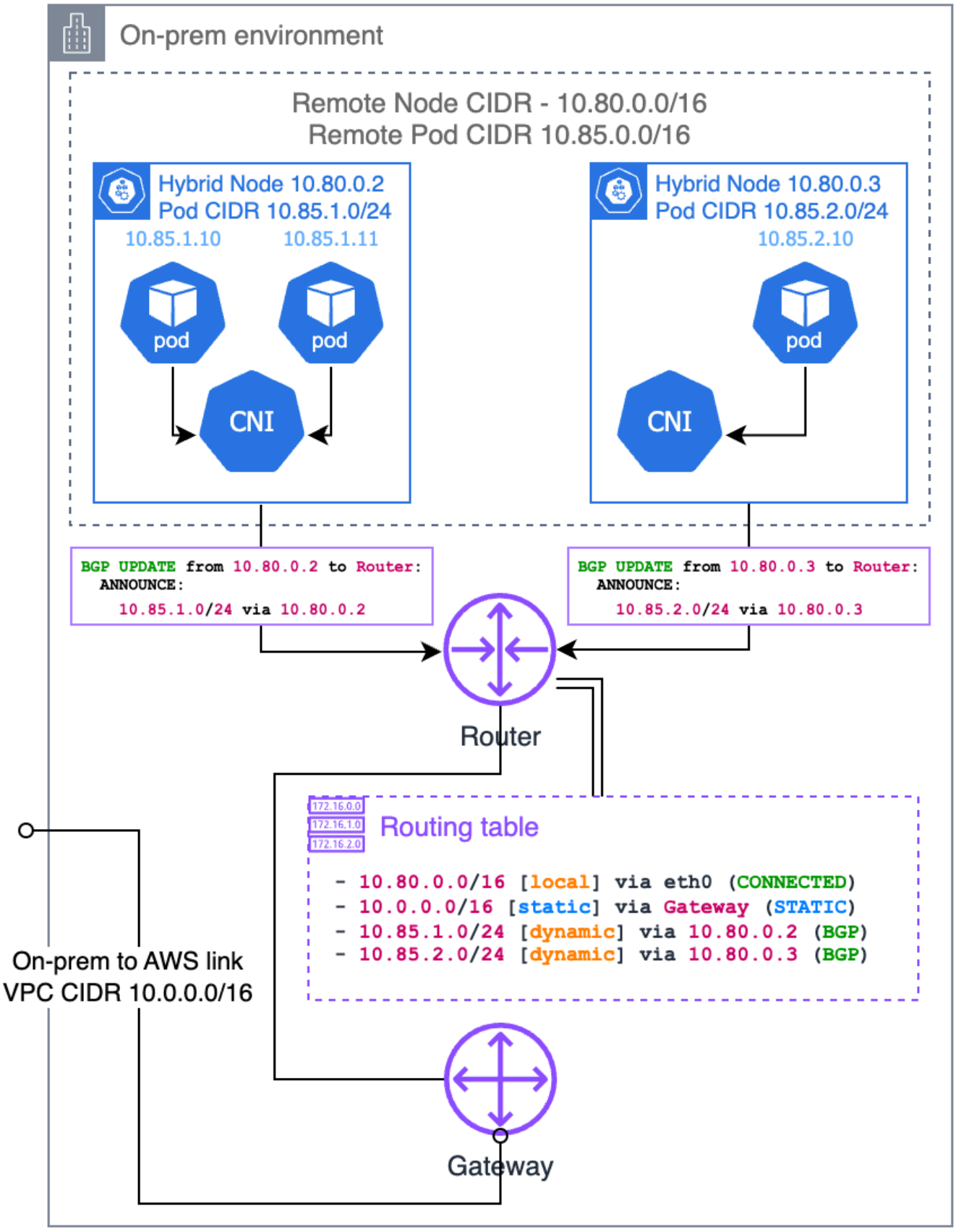
```
      name: default
    current-context: default
    users:
    - name: default
      user:
        tokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
kind: ConfigMap
metadata:
  name: kube-proxy
  namespace: kube-system
```

CIDR dei pod remoti instradabili

La pagina the section called "Concetti di rete" descrive in dettaglio i requisiti per eseguire webhook su nodi ibridi o per far sì che i pod in esecuzione su nodi cloud comunichino con quelli in esecuzione su nodi ibridi. Il requisito fondamentale è che il router on-premises debba sapere quale nodo è responsabile di un particolare pod IP. Esistono diversi modi per raggiungere questo obiettivo, tra cui Border Gateway Protocol (BGP), route statiche e proxy ARP (Address Resolution Protocol). Le diverse fasi vengono illustrate nelle sezioni seguenti.

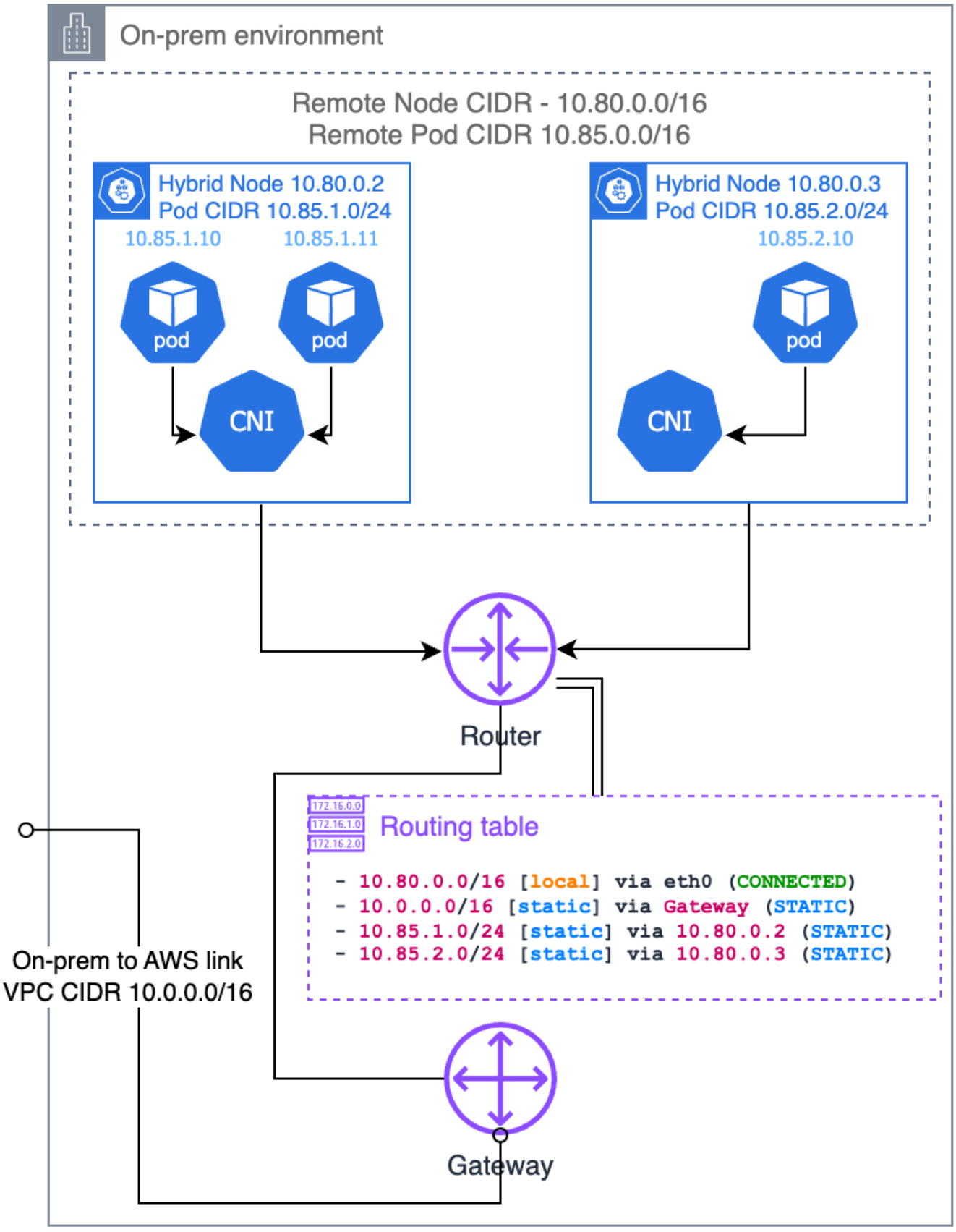
Border Gateway Protocol (BGP)

Se il tuo CNI lo supporta (come Cilium e Calico), puoi utilizzare la modalità BGP del tuo CNI per propagare le route ai tuoi pod CIDR per nodo dai nodi al router locale. Quando si utilizza la modalità BGP del CNI, quest'ultimo funge da router virtuale, quindi il router locale ritiene che il pod CIDR appartenga a una sottorete diversa e il nodo sia il gateway di accesso a quella sottorete.


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
BGP UPDATE from 10.80.0.2 to Router:
ANNOUNCE:
10.85.1.0/24 via 10.80.0.2
BGP UPDATE from 10.80.0.3 to Router:
ANNOUNCE:
10.85.2.0/24 via 10.80.0.3
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [dynamic] via 10.80.0.2 (BGP)
- 10.85.2.0/24 [dynamic] via 10.80.0.3 (BGP)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

### Route statiche

In alternativa, puoi configurare percorsi statici nel router locale. Questo è il modo più semplice per indirizzare il pod CIDR on-premises al tuo VPC, ma è anche il più soggetto a errori e difficile da gestire. È necessario assicurarsi che i percorsi siano sempre aggiornati con i nodi esistenti e i pod CIDR a loro assegnati. Se il numero di nodi è ridotto e l'infrastruttura è statica, questa è un'opzione valida e elimina la necessità del supporto BGP nel router. Se scegli questa opzione, ti consigliamo di configurare il tuo CNI con la slice CIDR del pod che desideri assegnare a ciascun nodo invece di lasciare che sia l'IPAM a decidere.


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [static] via 10.80.0.2 (STATIC)
- 10.85.2.0/24 [static] via 10.80.0.3 (STATIC)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

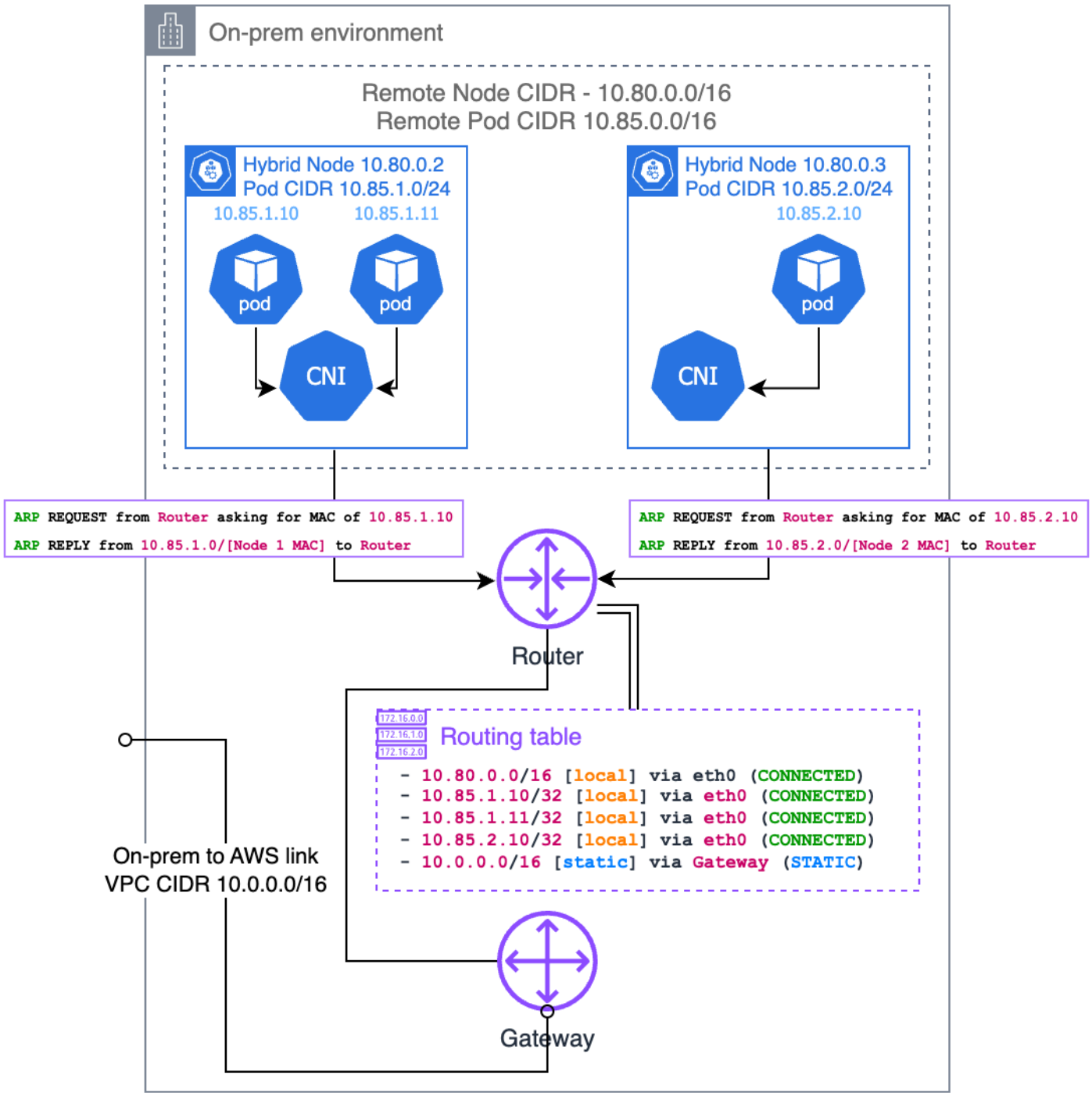
### Proxy ARP (Address Resolution Protocol)

Il proxy ARP è un altro approccio per rendere instradabili gli IP dei on-premises, particolarmente utile quando i nodi ibridi si trovano sulla stessa rete di livello 2 del router locale. Con il proxy ARP abilitato, un nodo risponde alle richieste ARP per gli IP dei pod che ospita, anche se tali IP appartengono a una sottorete diversa.

Quando un dispositivo sulla rete locale tenta di raggiungere un pod IP, invia innanzitutto una richiesta ARP che chiede "Chi ha questo IP?". Il nodo ibrido che ospita il pod risponderà con il proprio indirizzo MAC, dicendo "Posso gestire il traffico per quell'IP". Questo crea un percorso diretto tra i dispositivi della rete locale e i pod senza richiedere la configurazione del router.

Perché ciò funzioni, il CNI deve supportare la funzionalità proxy ARP. Cilium ha un supporto integrato per il proxy ARP che puoi abilitare tramite la configurazione. La considerazione fondamentale è che il pod CIDR non deve sovrapporsi a nessun'altra rete dell'ambiente, poiché ciò potrebbe causare conflitti di routing.

Questo approccio presenta diversi vantaggi: * Non è necessario configurare il router con BGP o mantenere percorsi statici * Funziona bene in ambienti in cui non si ha il controllo sulla configurazione del router




## Incapsulamento da pod a pod

Negli ambienti on-premises, i CNI utilizzano in genere protocolli di incapsulamento per creare reti overlay in grado di funzionare sulla rete fisica senza la necessità di riconfigurarla. Questa sezione spiega come funziona questo incapsulamento. Tieni presente che alcuni dettagli potrebbero variare a seconda del CNI che stai utilizzando.

L'incapsulamento avvolge i pacchetti della rete di pod originali all'interno di un altro pacchetto di rete che può essere instradato attraverso la rete fisica sottostante. Ciò consente ai pod di comunicare tra i nodi che eseguono lo stesso CNI senza che la rete fisica sappia come instradare quei pod CIDR.

Il protocollo di incapsulamento più comune utilizzato con Kubernetes è Virtual Extensible LAN (VXLAN), sebbene siano disponibili anche altri (come `Geneve`) a seconda del CNI utilizzato.

Incapsulamento VXLAN

VXLAN incapsula i frame Ethernet di livello 2 all'interno di pacchetti UDP. Quando un pod invia traffico a un altro pod su un nodo diverso, il CNI esegue le seguenti operazioni:

1. Il CNI intercetta i pacchetti dal Pod A.
2. Il CNI avvolge il pacchetto originale in un'intestazione VXLAN.
3. Questo pacchetto avvolto viene quindi inviato attraverso lo stack di rete normale del nodo al nodo di destinazione.
4. Il CNI sul nodo di destinazione scarta il pacchetto e lo consegna al Pod B.

Ecco cosa succede alla struttura dei pacchetti durante l'incapsulamento VXLAN:

Pacchetto originale da pod a pod:

```
+-----------------+---------------+-------------+-----------------+
| Ethernet Header | IP Header     | TCP/UDP     | Payload         |
| Src: Pod A MAC  | Src: Pod A IP | Src Port    |                 |
| Dst: Pod B MAC  | Dst: Pod B IP | Dst Port    |                 |
+-----------------+---------------+-------------+-----------------+
```

Dopo l'incapsulamento VXLAN

```
+-----------------+-------------+--------------+------------
+---------------------------+
| Outer Ethernet  | Outer IP    | Outer UDP    | VXLAN      | Original Pod-to-Pod
 |
| Src: Node A MAC | Src: Node A | Src: Random  | VNI: xx    | Packet (unchanged
 |
| Dst: Node B MAC | Dst: Node B | Dst: 4789    |            | from above)
 |
+-----------------+-------------+--------------+------------
+---------------------------+
```

Il VXLAN Network Identifier (VNI) distingue tra diverse reti di sovrapposizione.

Scenari di comunicazione dei pod

Pod sullo stesso nodo ibrido

Quando i pod sullo stesso nodo ibrido comunicano, in genere non è necessario alcun incapsulamento. Il CNI imposta percorsi locali che indirizzano il traffico tra i pod attraverso le interfacce virtuali interne del nodo:

```
Pod A -> veth0 -> node's bridge/routing table -> veth1 -> Pod B
```

Il pacchetto non lascia mai il nodo e non richiede l'incapsulamento.

Pod su diversi nodi ibridi

La comunicazione tra pod su diversi nodi ibridi richiede l'incapsulamento:

```
Pod A -> CNI -> [VXLAN encapsulation] -> Node A network -> router or gateway -> Node B
 network -> [VXLAN decapsulation] -> CNI -> Pod B
```

Ciò consente al traffico dei pod di attraversare l'infrastruttura di rete fisica senza che la rete fisica comprenda il routing IP dei pod.

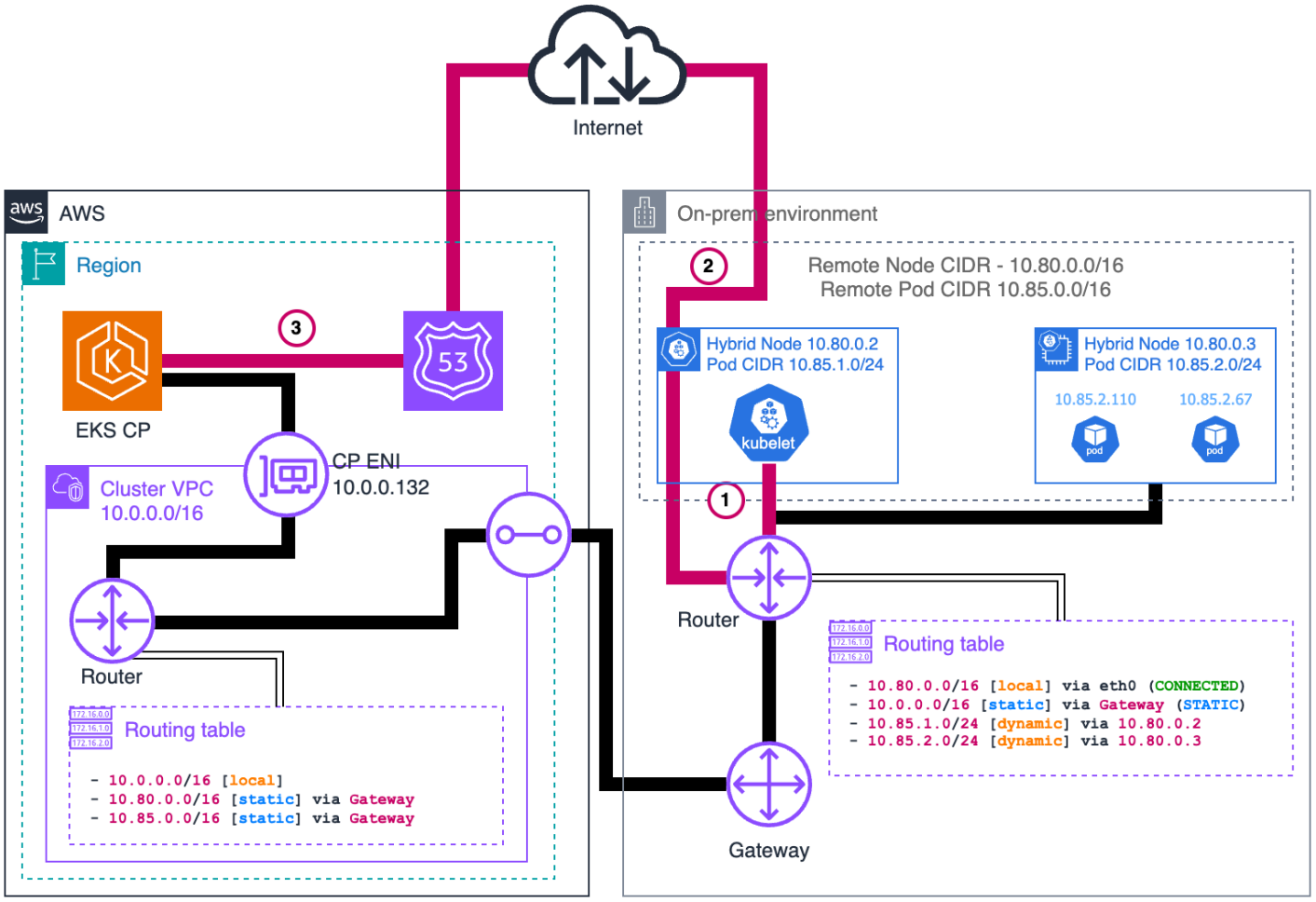
## Flussi di traffico di rete per nodi ibridi

Questa pagina descrive in dettaglio i flussi di traffico di rete per EKS Hybrid Nodes con diagrammi che mostrano i percorsi di rete end-to-end per i diversi tipi di traffico.

Sono coperti i seguenti flussi di traffico:

- the section called "Da nodo ibrido `kubelet` al piano di controllo EKS"
- the section called "Dal piano di controllo EKS al nodo ibrido (server `kubelet`)"
- the section called "I pod vengono eseguiti su nodi ibridi verso il piano di controllo (control-plane) EKS"
- the section called "Dal piano di controllo (control-plane) EKS ai pod in esecuzione su un nodo ibrido (webhook)"
- the section called "Pod-to-Pod in esecuzione su nodi ibridi"
- the section called "Da pod su nodi cloud a pod su nodi ibridi (traffico est-ovest)"

Da nodo ibrido **`kubelet` al piano** di controllo EKS




Richiesta

1**`kubelet`**. Avvia la richiesta

Quando il `kubelet` su un nodo ibrido deve comunicare con il piano di controllo (control-plane) EKS (ad esempio, per segnalare lo stato del nodo oppure ottenere le specifiche del pod), utilizza il file `kubeconfig` fornito durante la registrazione del nodo. Questo `kubeconfig` contiene l'URL dell'endpoint del server API (il nome DNS di Route53) anziché gli indirizzi IP diretti.

Il `kubelet` esegue una ricerca DNS per l'endpoint (ad esempio, `https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-west-2.eks.amazonaws.com`). In un cluster ad accesso pubblico, questo si risolve in un indirizzo IP pubblico (ad esempio, `54.239.118.52`) che appartiene al servizio EKS in esecuzione su AWS. Quindi, il `kubelet` crea una richiesta HTTPS sicura per questo endpoint. Il pacchetto iniziale appare come riportato di seguito:

```
+-------------------+--------------------+-----------------+
| IP Header         | TCP Header         | Payload         |
| Src: 10.80.0.2    | Src: 52390 (random) |                 |
| Dst: 54.239.118.52 | Dst: 443           |                 |
+-------------------+--------------------+-----------------+
```

2. Routing del router locale

Poiché l'IP di destinazione è un indirizzo IP pubblico e non fa parte della rete locale, il `kubelet` invia questo pacchetto al gateway predefinito (il router locale on-premises). Il router esamina l'IP di destinazione e determina che si tratta di un indirizzo IP pubblico.

Per il traffico pubblico, il router in genere inoltra il pacchetto a un gateway Internet o a un router di confine che gestisce il traffico in uscita verso internet. Questo valore è omesso nel diagramma e dipenderà da come è configurata la rete on-premises. Il pacchetto attraversa l'infrastruttura di rete on-premises e alla fine raggiunge la rete del provider di servizi internet.

3. Distribuzione al piano di controllo (control-plane) EKS

Il pacchetto viaggia attraverso la rete Internet pubblica e le reti di transito fino a raggiungere AWS la rete. AWS La rete indirizza il pacchetto all'endpoint del servizio EKS nella regione appropriata. Quando il pacchetto raggiunge il servizio EKS, viene inoltrato all'effettivo piano di controllo (control-plane) EKS del cluster.

Questo routing attraverso la rete Internet pubblica è diverso dal VPC-routed percorso privato che vedremo in altri flussi di traffico. La differenza fondamentale è che quando si utilizza la modalità di accesso pubblico, il traffico dal `kubelet` on-premises (ma non dai pod) al piano di controllo (control-plane) EKS non passa attraverso il VPC, ma utilizza invece l'infrastruttura internet globale.

Risposta

Dopo che il piano di controllo (control-plane) EKS ha elaborato la richiesta del `kubelet`, invia una risposta:

3. Il piano di controllo (control-plane) EKS invia una risposta

Il piano di controllo (control-plane) EKS crea un pacchetto di risposta. Questo pacchetto ha l'IP pubblico come origine e l'IP del nodo ibrido come destinazione:

```
+-------------------+--------------------+-----------------+
| IP Header         | TCP Header         | Payload         |
```

```
| Src: 54.239.118.52 | Src: 443           |                 |
| Dst: 10.80.0.2     | Dst: 52390         |                 |
+--------------------+--------------------+-----------------+
```

2. Routing internet

Il pacchetto di risposta attraversa internet, seguendo il percorso di routing determinato dai provider di servizi internet, fino a raggiungere il router edge della rete on-premises.

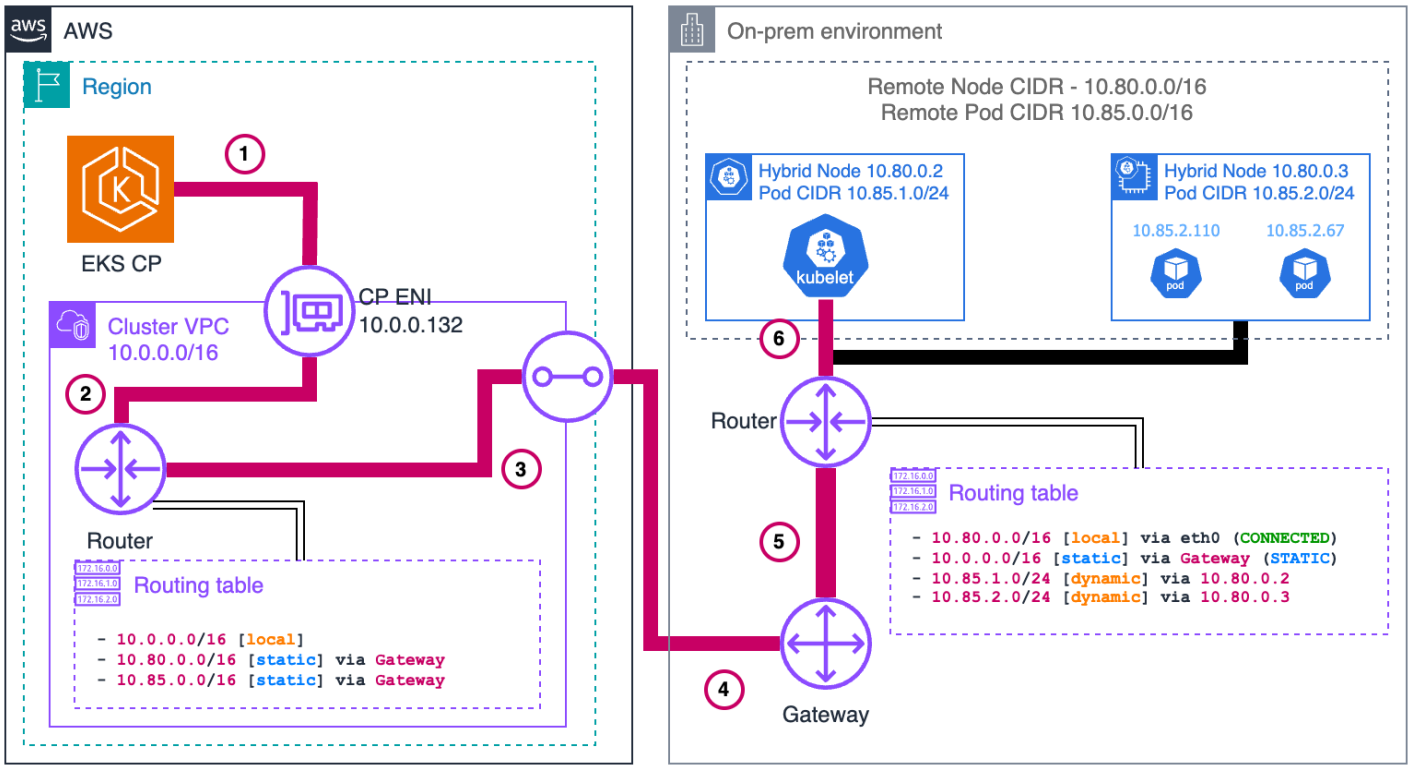
1. Distribuzione locale

Il router on-premises riceve il pacchetto e riconosce l'IP di destinazione (`10.80.0.2`) come appartenente alla rete locale. Inoltra il pacchetto attraverso l'infrastruttura di rete locale fino a raggiungere il nodo ibrido di destinazione, dove il `kubelet` riceve ed elabora la risposta.

Da nodo ibrido **`kube-proxy`** al piano di controllo EKS

Se abiliti l'accesso pubblico agli endpoint per il cluster, il traffico di ritorno utilizza la rete internet pubblica. Questo traffico proviene dal nodo ibrido verso il `kube-proxy` piano di controllo EKS e segue lo stesso percorso del traffico dal nodo ibrido `kubelet` al piano di controllo EKS.

Dal piano di controllo EKS al nodo ibrido (server **`kubelet`**)



Richiesta

1. Il server API EKS Kubernetes avvia la richiesta

Il server API EKS Kubernetes recupera l'indirizzo IP del nodo (`10.80.0.2`) dallo stato dell'oggetto del nodo. Quindi indirizza questa richiesta tramite la sua interfaccia di rete elastica (ENI) nel VPC, poiché l'IP di destinazione appartiene al CIDR del nodo remoto configurato (`10.80.0.0/16`). Il pacchetto iniziale appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 67493 (random) |                 |
| Dst: 10.80.0.2  | Dst: 10250          |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione di rete del VPC

Il pacchetto lascia l'ENI ed entra nel livello di rete del VPC, dove viene diretto al gateway della sottorete per un ulteriore instradamento.

3. Ricerca della tabella di routing del VPC

La tabella di routing del VPC per la sottorete contenente l'ENI del piano di controllo (control-plane) EKS ha un instradamento specifico (il secondo nel diagramma) per il CIDR del nodo remoto. In base a questa regola di routing, il pacchetto viene indirizzato al VPC-to-onprem gateway.

4. Cross-boundary transito

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises.

5. On-premises ricezione in rete

Il pacchetto arriva al router on-premises locale che gestisce il traffico per la sottorete in cui si trovano i nodi ibridi.

6. Distribuzione finale

Il router locale identifica che l'indirizzo IP (`10.80.0.2`) di destinazione appartiene alla sua rete connessa direttamente e inoltra il pacchetto al nodo ibrido di destinazione, dove il `kubelet` riceve ed elabora la richiesta.

Risposta

Dopo che il `kubelet` del nodo ibrido ha elaborato la richiesta, restituisce una risposta seguendo lo stesso percorso in senso inverso:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 10250          |                 |
| Dst: 10.0.0.132 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

6. **`kubelet`** Invia la risposta

Il `kubelet` sul nodo ibrido (`10.80.0.2`) crea un pacchetto di risposta con l'IP di origine originale come destinazione. La destinazione non appartiene alla rete locale, quindi viene inviata al gateway predefinito dell'host, che è il router locale.

5. Routing del router locale

Il router determina l'appartenenza a `10.0.0.0/16` dell'IP di destinazione (`10.0.0.132`), che dispone di un instradamento che punta al gateway a cui si connette AWS.

4. Cross-Boundary Ritorno

Il pacchetto torna indietro attraverso la stessa connessione on-premises al VPC (come Direct Connect o VPN), superando il limite del cloud nella direzione opposta.

3. Routing del VPC

Quando il pacchetto arriva nel VPC, le tabelle di routing identificano che l'IP di destinazione appartiene a un CIDR del VPC. L'instradamento dei pacchetti all'interno del VPC.

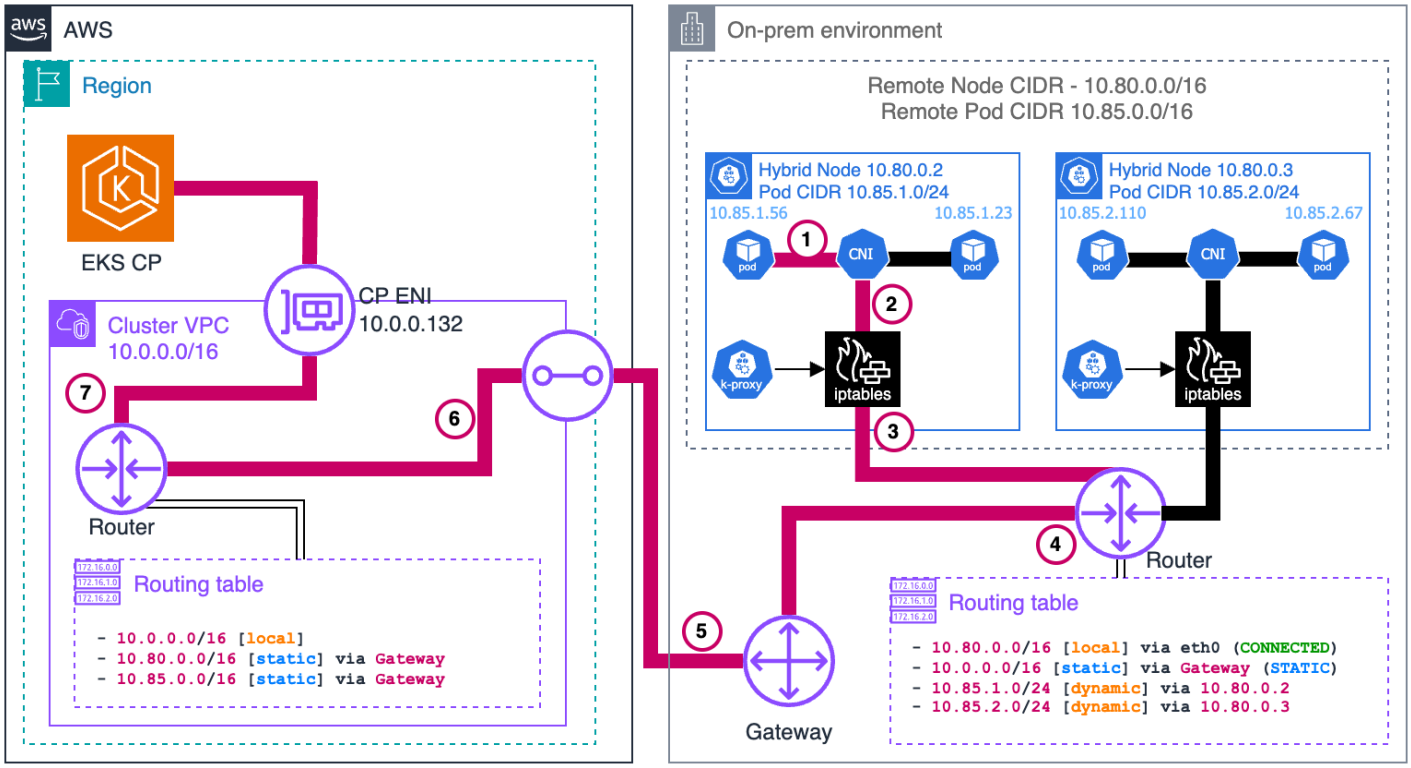
2. Distribuzione di rete del VPC

Il livello di rete del VPC inoltra il pacchetto alla sottorete con l'ENI del piano di controllo (control-plane) EKS (`10.0.0.132`).

1. Ricezione dell'ENI

Il pacchetto raggiunge l'ENI del piano di controllo (control-plane) EKS collegata al server API Kubernetes, completando il viaggio di andata e ritorno.

I pod vengono eseguiti su nodi ibridi verso il piano di controllo (control-plane) EKS




Senza NAT della CNI

Richiesta

In genere, i pod comunicano al server API Kubernetes tramite il servizio `kubernetes`. L'IP del servizio è il primo IP del servizio CIDR del cluster. Questa convenzione consente ai pod che devono essere eseguiti prima che CoreDNS sia disponibile per raggiungere il server API, ad esempio la Container Network Interface (CNI). Le richieste lasciano il pod con l'IP del servizio come destinazione. Ad esempio, se il servizio CIDR è `172.16.0.0/16`, l'IP del servizio è `172.16.0.1`.

1. Il pod avvia la richiesta

Il pod invia una richiesta all'IP del servizio `kubernetes` (`172.16.0.1`) sulla porta del server API (443) da una porta di origine casuale. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione della CNI

La CNI rileva che l'IP di destinazione non appartiene a nessun CIDR del pod che gestisce. Poiché il NAT in uscita è disabilitato, la CNI passa il pacchetto allo stack di rete host senza modificarlo.

3. Elaborazione della rete del nodo

Il pacchetto entra nello stack di rete del nodo dove gli hook `netfilter` attivano le regole `iptables` impostate da kube-proxy. Diverse regole si applicano nell'ordine seguente:

1. Il pacchetto colpisce per primo la catena `KUBE-SERVICES`, che contiene regole che corrispondono al ClusterIP e alla porta di ogni servizio.
2. La regola di corrispondenza passa alla catena `KUBE-SVC-XXX` del servizio `kubernetes` (pacchetti destinati a `172.16.0.1:443`), che contiene le regole di bilanciamento del carico.
3. La regola di bilanciamento del carico seleziona casualmente una delle catene `KUBE-SEP-XXX` per gli IP del piano di controllo (control-plane) dell'ENI (`10.0.0.132` o `10.0.1.23`).
4. La catena `KUBE-SEP-XXX` selezionata ha la regola effettiva che modifica l'IP di destinazione dall'IP del servizio a quello selezionato. Si chiama Destination Network Address Translation (DNAT).

Dopo aver applicato queste regole, supponendo che l'IP dell'ENI del piano di controllo (control-plane) EKS selezionato sia `10.0.0.132`, il pacchetto appare come indicato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Il nodo inoltra il pacchetto al gateway predefinito perché l'IP di destinazione non si trova nella rete locale.

4. Routing del router locale

Il router locale determina che l'IP di destinazione (`10.0.0.132`) appartiene al CIDR del VPC (`10.0.0.0/16`) e lo inoltra al gateway che si connette ad AWS.

5. Cross-Boundary Transito

Il pacchetto viaggia attraverso la connessione stabilita (come Direct Connect o VPN) oltre il limite del cloud fino al VPC.

6. Distribuzione di rete del VPC

Il livello di rete del VPC indirizza il pacchetto alla sottorete corretta in cui si trova l'ENI del piano di controllo (control-plane) EKS (`10.0.0.132`).

7. Ricezione dell'ENI

Il pacchetto raggiunge l'ENI del piano di controllo (control-plane) EKS collegato al server API Kubernetes.

Risposta

Dopo che il piano di controllo (control-plane) EKS ha elaborato la richiesta, invia una risposta al pod:

7. Il server API invia una risposta

Il server API EKS Kubernetes crea un pacchetto di risposta con l'IP di origine originale come destinazione. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Poiché l'IP di destinazione appartiene al CIDR del pod remoto configurato (`10.85.0.0/16`), lo invia tramite la sua ENI nel VPC con il router della sottorete come hop successivo.

6. Routing del VPC

La tabella di routing VPC contiene una voce per il pod remoto CIDR (`10.85.0.0/16`), che indirizza questo traffico verso il gateway. VPC-to-onprem

5. Cross-Boundary Transito

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises.

4. On-Premises Ricezione in rete

Il pacchetto arriva al router on-premises locale.

3. Distribuzione al nodo

La tabella del router contiene una voce per `10.85.1.0/24` con `10.80.0.2` come hop successivo, che consegna il pacchetto al nostro nodo.

2. Elaborazione della rete del nodo

Poiché il pacchetto viene elaborato dallo stack di rete del nodo, `conntrack` (una parte di `netfilter`) abbina il pacchetto alla connessione inizialmente stabilita dal pod. Poiché il DNAT è stato originariamente applicato, `conntrack` inverte il DNAT riscrivendo l'IP di origine dall'IP dell'ENI del piano di controllo (control-plane) EKS all'IP del servizio `kubernetes`:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Elaborazione della CNI

La CNI identifica che l'IP di destinazione appartiene a un pod nella sua rete e consegna il pacchetto al namespace della rete del pod corretto.

Questo flusso mostra perché i CIDR dei pod remoti devono essere instradati correttamente dal VPC fino al nodo specifico che ospita ciascun pod: l'intero percorso di ritorno dipende dal corretto instradamento degli IP del pod su reti cloud e on-premises.

Con NAT della CNI

Questo flusso è molto simile a quello senza NAT della CNI, ma con una differenza fondamentale: la CNI applica il NAT di origine (SNAT) al pacchetto prima di inviarlo allo stack di rete del nodo. Ciò modifica l'IP di origine del pacchetto nell'IP del nodo, permettendo al pacchetto di essere reindirizzato al nodo senza richiedere una configurazione di routing aggiuntiva.

Richiesta

1. Il pod avvia la richiesta

Il pod invia una richiesta all'IP del servizio `kubernetes` (`172.16.0.1`) sulla porta del server API EKS Kubernetes (443) da una porta di origine casuale. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione della CNI

La CNI rileva che l'IP di destinazione non appartiene a nessun CIDR del pod che gestisce. Poiché il NAT in uscita è abilitato, la CNI applica lo SNAT al pacchetto, cambiando l'IP di origine con l'IP del nodo prima di passarlo allo stack di rete del nodo:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Nota: nell'esempio la CNI e `iptables` sono mostrati come blocchi separati per motivi di chiarezza, ma in pratica è possibile che alcune CNI utilizzino `iptables` per applicare il NAT.

3. Elaborazione della rete del nodo

Qui le regole `iptables` stabilite da `kube-proxy` si comportano come nell'esempio precedente, ovvero il bilanciamento del carico del pacchetto su una delle ENI del piano di controllo (control-plane) EKS. Il pacchetto ora appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Il nodo inoltra il pacchetto al gateway predefinito perché l'IP di destinazione non si trova nella rete locale.

4. Routing del router locale

Il router locale determina che l'IP di destinazione (`10.0.0.132`) appartiene al CIDR del VPC (`10.0.0.0/16`) e lo inoltra al gateway che si connette ad AWS.

5. Cross-Boundary Transito

Il pacchetto viaggia attraverso la connessione stabilita (come Direct Connect o VPN) oltre il limite del cloud fino al VPC.

6. Distribuzione di rete del VPC

Il livello di rete del VPC indirizza il pacchetto alla sottorete corretta in cui si trova l'ENI del piano di controllo (control-plane) EKS (`10.0.0.132`).

7. Ricezione dell'ENI

Il pacchetto raggiunge l'ENI del piano di controllo (control-plane) EKS collegato al server API Kubernetes.

Risposta

Dopo che il piano di controllo (control-plane) EKS ha elaborato la richiesta, invia una risposta al pod:

7. Il server API invia una risposta

Il server API EKS Kubernetes crea un pacchetto di risposta con l'IP di origine originale come destinazione. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Poiché l'IP di destinazione appartiene al CIDR del nodo remoto configurato (`10.80.0.0/16`), lo invia tramite la sua ENI nel VPC con il router della sottorete come hop successivo.

6. Routing del VPC

La tabella di routing VPC contiene una voce per il nodo remoto CIDR (`10.80.0.0/16`), che indirizza questo traffico verso il gateway. VPC-to-onprem

5. Cross-Boundary Transito

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises.

4. On-Premises Ricezione in rete

Il pacchetto arriva al router on-premises locale.

3. Distribuzione al nodo

Il router locale identifica che l'indirizzo IP (`10.80.0.2`) di destinazione appartiene alla sua rete connessa direttamente e inoltra il pacchetto al nodo ibrido di destinazione.

2. Elaborazione della rete del nodo

Poiché il pacchetto viene elaborato dallo stack di rete del nodo, `conntrack` (una parte di `netfilter`) abbina il pacchetto alla connessione stabilita inizialmente dal pod e, poiché il DNAT è stato originariamente applicato, inverte la situazione riscrivendo l'IP di origine dall'IP dell'ENI del piano di controllo (control-plane) EKS a quello del servizio `kubernetes`:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Elaborazione della CNI

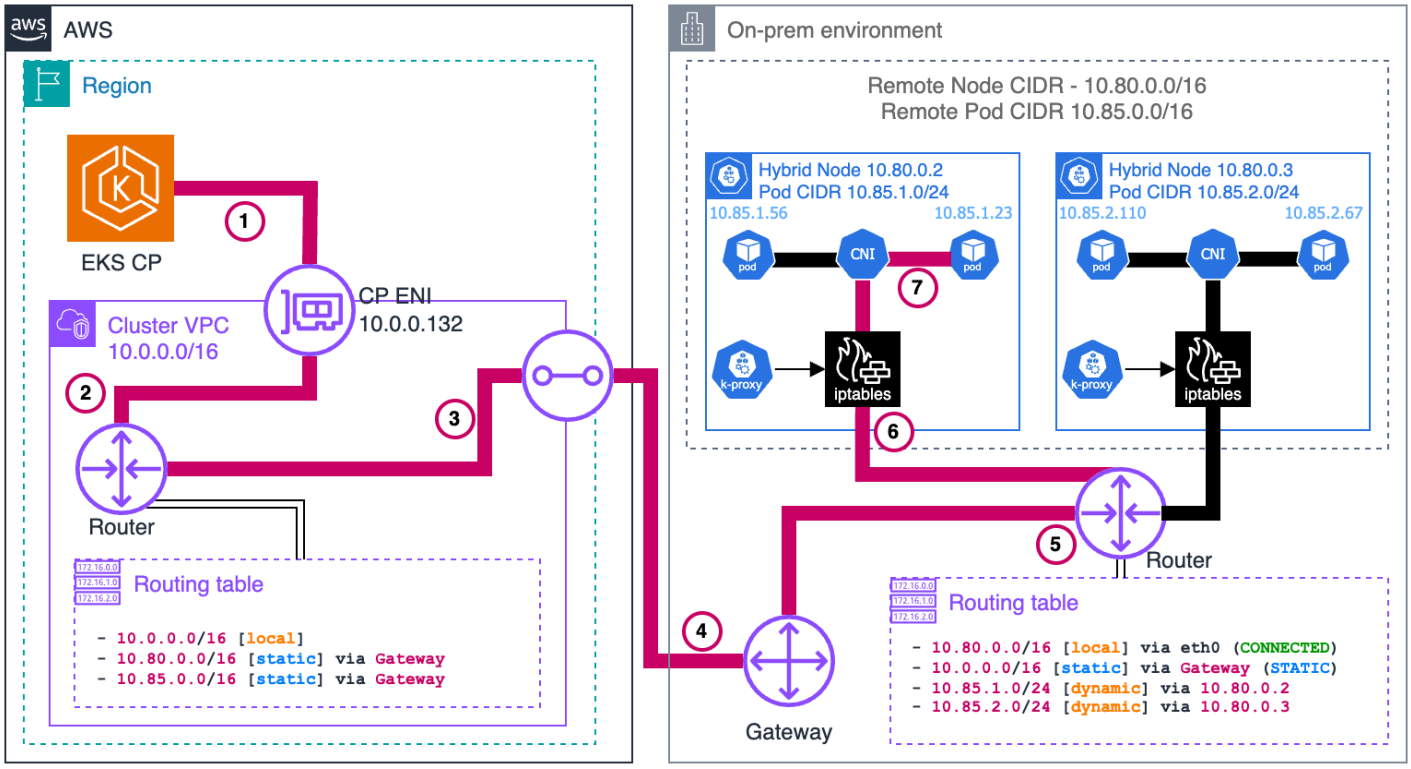
Il CNI identifica che questo pacchetto appartiene a una connessione a cui ha precedentemente applicato lo SNAT. Inverte lo SNAT, riportando l'IP di destinazione all'IP del pod:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

La CNI identifica che l'IP di destinazione appartiene a un pod nella sua rete e consegna il pacchetto al namespace della rete del pod corretto.

Questo flusso mostra come CNI NAT-ing può semplificare la configurazione consentendo il reindirizzamento dei pacchetti al nodo senza richiedere un routing aggiuntivo per i pod CIDR.

Dal piano di controllo (control-plane) EKS ai pod in esecuzione su un nodo ibrido (webhook)




Questo schema di traffico si riscontra più comunemente con i webhook, in cui il piano di controllo (control-plane) EKS deve avviare direttamente le connessioni ai server webhook in esecuzione nei pod su nodi ibridi. Gli esempi includono la convalida e la modifica dei webhook di ammissione, che vengono chiamati dal server API durante i processi di convalida o mutazione delle risorse.

Richiesta

1. Il server API EKS Kubernetes avvia la richiesta

Quando un webhook è configurato nel cluster e un'operazione API pertinente lo attiva, il server API EKS Kubernetes deve stabilire una connessione diretta al pod del server webhook. Il server API cerca innanzitutto l'indirizzo IP del pod dalla risorsa Service o Endpoint associata al webhook.

Supponendo che il pod del webhook sia in esecuzione su un nodo ibrido con IP `10.85.1.23`, il server API EKS Kubernetes crea una richiesta HTTPS all'endpoint del webhook. Il pacchetto iniziale viene inviato tramite l'ENI del piano di controllo (control-plane) EKS nel VPC perché l'IP di destinazione `10.85.1.23` appartiene al CIDR del pod remoto configurato (`10.85.0.0/16`). Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
```

```
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 41892 (random) |                 |
| Dst: 10.85.1.23 | Dst: 8443           |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione di rete del VPC

Il pacchetto lascia l'ENI del piano di controllo (control-plane) EKS ed entra nel livello di rete del VPC con il router della sottorete come hop successivo.

3. Ricerca della tabella di routing del VPC

La tabella di routing del VPC per la sottorete contenente l'ENI del piano di controllo (control-plane) EKS contiene un instradamento specifico per il CIDR del pod remoto (`10.85.0.0/16`). Questa regola di routing indirizza il pacchetto VPC-to-onprem verso il gateway (ad esempio, un Virtual Private Gateway per Direct Connect o connessioni VPN):

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

4. Cross-Boundary Transito

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises. Il pacchetto mantiene gli indirizzi IP di origine e destinazione originali mentre attraversa questa connessione.

5. On-Premises Ricezione in rete

Il pacchetto arriva al router on-premises locale. Il router consulta la propria tabella di routing per determinare come raggiungere l'indirizzo 10.85.1.23. Perché ciò funzioni, la rete on-premises deve disporre di percorsi per i CIDR dei pod che indirizzano il traffico verso il nodo ibrido appropriato.

In questo caso, la tabella di routing del router contiene una voce che indica che la sottorete `10.85.1.0/24` è raggiungibile tramite il nodo ibrido con IP `10.80.0.2`:

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

6. Distribuzione al nodo

In base alla voce della tabella di routing, il router inoltra il pacchetto al nodo ibrido (`10.80.0.2`). Quando il pacchetto arriva al nodo, ha lo stesso aspetto di quando il server API EKS Kubernetes lo ha inviato, con l'IP di destinazione che è ancora l'IP del pod.

7. Elaborazione della CNI

Lo stack di rete del nodo riceve il pacchetto e, dato che l'IP di destinazione non è quello del nodo, lo passa alla CNI per l'elaborazione. La CNI identifica che l'IP di destinazione appartiene a un pod in esecuzione localmente su questo nodo e inoltra il pacchetto al pod corretto tramite le interfacce virtuali appropriate:

```
Original packet -> node routing -> CNI -> Pod's network namespace
```

Il server webhook nel pod riceve la richiesta e la elabora.

Risposta

Dopo che il pod del webhook ha elaborato la richiesta, restituisce una risposta seguendo lo stesso percorso in senso inverso:

7. Il pod invia la risposta

Il pod del webhook crea un pacchetto di risposta con il proprio IP come origine e il richiedente originale (l'ENI del piano di controllo (control-plane) EKS) come destinazione:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.23 | Src: 8443           |                 |
| Dst: 10.0.0.132 | Dst: 41892          |                 |
+-----------------+---------------------+-----------------+
```

La CNI identifica che questo pacchetto va a una rete esterna (non a un pod locale) e lo passa allo stack di rete del nodo mantenendo l'IP di origine originale.

6. Elaborazione della rete del nodo

Il nodo determina che l'IP di destinazione (`10.0.0.132`) non si trova nella rete locale e inoltra il pacchetto al gateway predefinito (il router locale).

5. Routing del router locale

Il router locale consulta la sua tabella di routing e determina che l'IP di destinazione (`10.0.0.132`) appartiene al CIDR del VPC (`10.0.0.0/16`). Inoltra il pacchetto al gateway che si connette a AWS.

4. Cross-Boundary Transito

Il pacchetto torna indietro attraverso la stessa connessione on-premises al VPC, superando il limite del cloud nella direzione opposta.

3. Routing del VPC

Quando il pacchetto arriva nel VPC, le tabelle di routing identificano che l'IP di destinazione appartiene a una sottorete all'interno del VPC. Il pacchetto viene instradato di conseguenza all'interno del VPC.

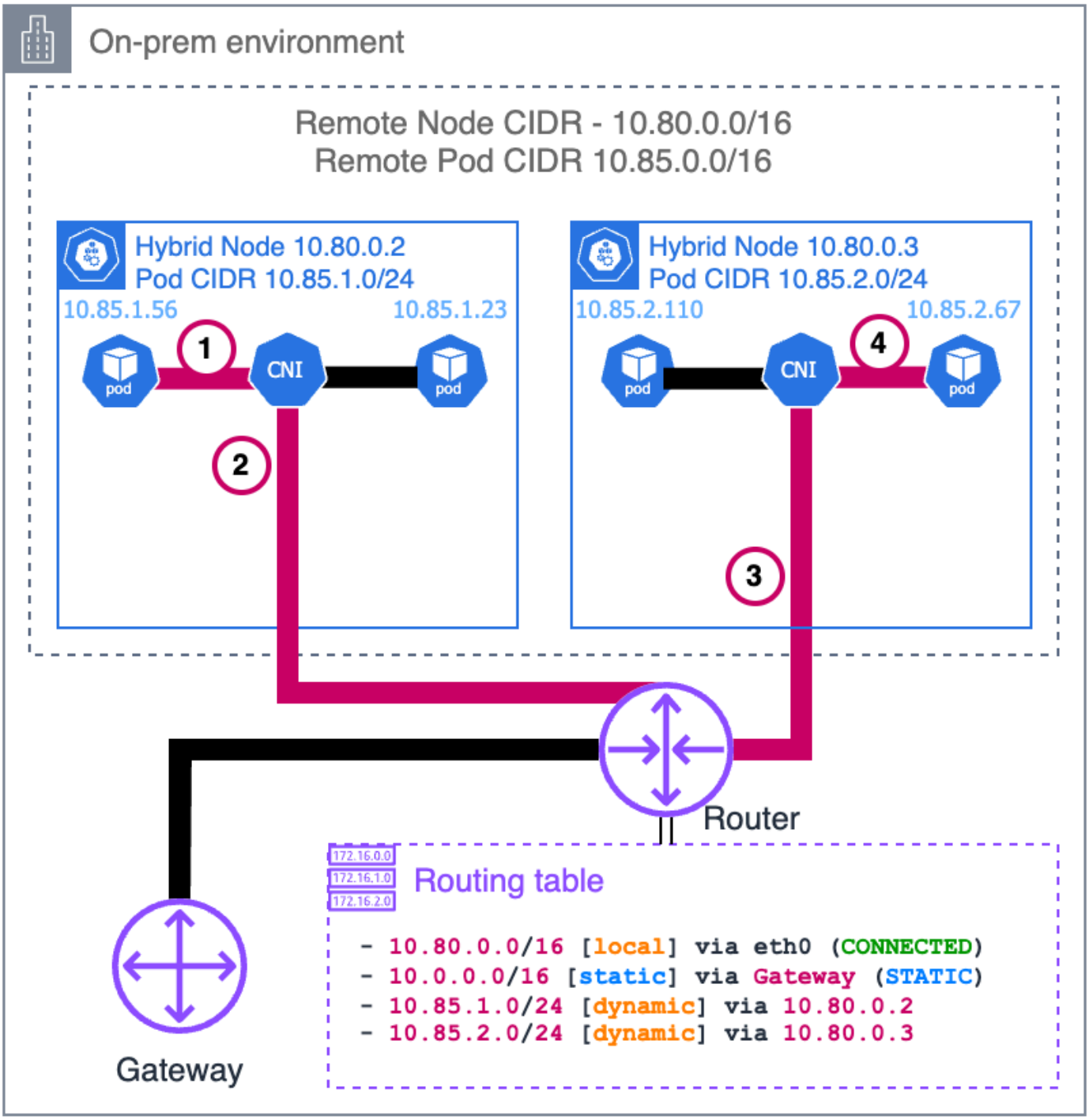
2. e 1. Ricezione dell'ENI del piano di controllo (control-plane) EKS

Il pacchetto raggiunge l'ENI collegata al server API Kubernetes, completando il viaggio di andata e ritorno. Il server API riceve la risposta del webhook e continua a elaborare la richiesta API originale in base a tale risposta.

Questo flusso di traffico dimostra perché i CIDR dei pod remoti devono essere configurati e instradati correttamente:

- Il VPC deve disporre di percorsi per i CIDR dei pod remoti che puntano al gateway on-premises
- La rete on-premises deve disporre di percorsi per i CIDR dei pod che indirizzano il traffico verso i nodi specifici che ospitano tali pod
- Senza questa configurazione di routing, i webhook e altri servizi simili eseguiti in pod su nodi ibridi non sarebbero raggiungibili dal piano di controllo (control-plane) EKS.

Pod-to-Pod in esecuzione su nodi ibridi




Questa sezione spiega come i pod in esecuzione su diversi nodi ibridi comunicano tra loro. Questo esempio presuppone che la CNI utilizzi VXLAN per l'incapsulamento, una caratteristica comune per

le CNI come Cilium o Calico. Il processo complessivo è simile per altri protocolli di incapsulamento come Geneve o. IP-in-IP

Richiesta

1. Il pod A avvia la comunicazione

Il pod A (`10.85.1.56`) sul nodo 1 desidera inviare traffico al pod B (`10.85.2.67`) sul nodo 2. Il pacchetto iniziale appare come riportato di seguito:

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
+------------------+-----------------+-------------+-----------------+
```

2. La CNI intercetta ed elabora il pacchetto

Quando il pacchetto del pod A lascia il suo namespace di rete, la CNI lo intercetta. La CNI consulta la sua tabella di routing e determina i seguenti elementi: - L'IP di destinazione (`10.85.2.67`) appartiene al CIDR del pod - Questo IP non si trova sul nodo locale ma appartiene al nodo 2 (`10.80.0.3`) - Il pacchetto deve essere incapsulato con VXLAN.

La decisione di incapsulare è fondamentale perché la rete fisica sottostante non sa come instradare direttamente i CIDR dei pod, ma solo come instradare il traffico tra gli IP dei nodi.

La CNI incapsula l'intero pacchetto originale all'interno di un frame VXLAN. Questo crea effettivamente un "pacchetto all'interno di un pacchetto" con nuovi header:

```
+-----------------+---------------+-------------+-----------
+--------------------------+
| Outer Ethernet  | Outer IP      | Outer UDP   | VXLAN      | Original Pod-to-Pod
    |
| Src: Node1 MAC  | Src: 10.80.0.2 | Src: Random  | VNI: 42    | Packet (unchanged
    |
| Dst: Router MAC | Dst: 10.80.0.3 | Dst: 8472    |            | from above)
    |
+-----------------+---------------+-------------+-----------
+--------------------------+
```

Punti chiave di questo incapsulamento: - Il pacchetto esterno viene indirizzato dal nodo 1 (`10.80.0.2`) al nodo 2 (`10.80.0.3`) - La porta UDP `8472` è la porta VXLAN utilizzata da Cilium

per impostazione predefinita - Il VXLAN Network Identifier (VNI) identifica a quale rete sovrapposta appartiene tale pacchetto - L'intero pacchetto originale (con l'IP del pod A come origine e l'IP del pod B come destinazione) è conservato intatto all'interno

Il pacchetto incapsulato entra ora nel normale stack di rete del nodo 1 e viene elaborato allo stesso modo di qualsiasi altro pacchetto:

1. Elaborazione della rete del nodo: lo stack di rete del nodo 1 indirizza il pacchetto in base alla sua destinazione (`10.80.0.3`)
2. Distribuzione della rete locale:
   - Se entrambi i nodi si trovano sulla stessa rete di livello 2, il pacchetto viene inviato direttamente al nodo 2
   - Se si trovano su sottoreti diverse, il pacchetto viene prima inoltrato al router locale
3. Gestione del router: il router inoltra il pacchetto in base alla sua tabella di routing, distribuendolo al nodo 2

3. Elaborazione dei nodi di ricezione

Quando il pacchetto incapsulato arriva al nodo 2 (`10.80.0.3`):

1. Lo stack di rete del nodo lo riceve e lo identifica come pacchetto VXLAN (porta UDP `4789`)
2. Il pacchetto viene passato all'interfaccia VXLAN della CNI per l'elaborazione

4. Decapsulazione VXLAN

La CNI sul nodo 2 elabora il pacchetto VXLAN:

1. Elimina gli header esterni (Ethernet, IP, UDP e VXLAN).
2. Estrae il pacchetto interno originale.
3. Il pacchetto è ora tornato alla sua forma originale:

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
+------------------+-----------------+-------------+-----------------+
```

La CNI sul nodo 2 esamina l'IP di destinazione (`10.85.2.67`) e:

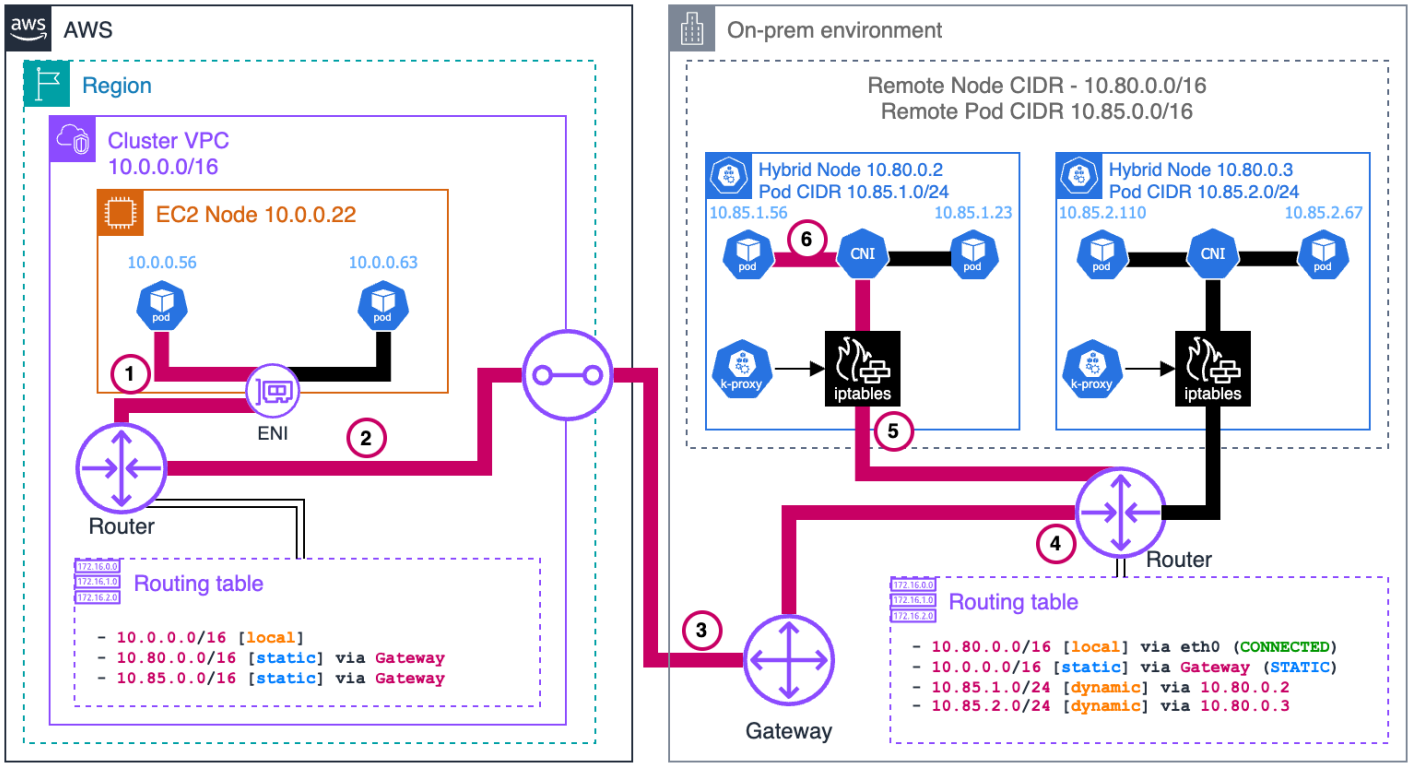
1. Identifica che questo IP appartiene a un pod locale.
2. Instrada il pacchetto attraverso le interfacce virtuali appropriate.
3. Fornisce il pacchetto al namespace di rete del pod B.

### Risposta

Quando il pod B risponde al pod A, l'intero processo avviene al contrario:

4. Il pod B invia un pacchetto al pod A (`10.85.1.56`).
5. La CNI del nodo 2 lo incapsula con VXLAN, impostando la destinazione sul nodo 1 (`10.80.0.2`).
6. Il pacchetto incapsulato viene consegnato al nodo 1.
7. La CNI del nodo 1 lo decapsula e fornisce la risposta originale al pod A.

### Da pod su nodi cloud a pod su nodi ibridi (traffico est-ovest)




### Richiesta

1. Il pod A avvia la comunicazione

Il pod A (`10.0.0.56`) sul nodo EC2 desidera inviare traffico al pod B (`10.85.1.56`) sul nodo ibrido. Il pacchetto iniziale appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.56  | Src: 52390 (random) |                 |
| Dst: 10.85.1.56 | Dst: 8080           |                 |
+-----------------+---------------------+-----------------+
```

Con la CNI del VPC, il pod A ha un IP dal CIDR del VPC ed è collegato direttamente a una ENI sull'istanza EC2. Il namespace di rete del pod è connesso alla rete del VPC, quindi il pacchetto entra direttamente nell'infrastruttura di routing VPC.

2. Routing del VPC

La tabella di routing VPC contiene una route specifica per il Remote Pod CIDR (`10.85.0.0/16`), che indirizza questo traffico verso il gateway: VPC-to-onprem

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

In base a questa regola di routing, il pacchetto viene indirizzato verso il gateway che si connette alla rete on-premises.

3. Cross-Boundary Transito

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises. Il pacchetto mantiene gli indirizzi IP di origine e di destinazione originali durante questo transito.

4. On-Premises Ricezione in rete

Il pacchetto arriva al router on-premises locale. Il router consulta la propria tabella di routing per determinare l'hop successivo per raggiungere l'indirizzo 10.85.1.56. Il router on-premises deve disporre di percorsi per i CIDR dei pod che indirizzano il traffico verso il nodo ibrido appropriato.

La tabella del router contiene una voce che indica che la sottorete `10.85.1.0/24` è raggiungibile tramite il nodo ibrido con IP `10.80.0.2`:

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

5. Elaborazione della rete del nodo

Il router inoltra il pacchetto al nodo ibrido (`10.80.0.2`). Quando il pacchetto arriva al nodo, ha ancora l'IP del pod A come origine e l'IP del pod B come destinazione.

6. Elaborazione della CNI

Lo stack di rete del nodo riceve il pacchetto e, visto che l'IP di destinazione non è il proprio, lo passa alla CNI per l'elaborazione. La CNI identifica che l'IP di destinazione appartiene a un pod in esecuzione localmente su questo nodo e inoltra il pacchetto al pod corretto tramite le interfacce virtuali appropriate:

```
Original packet -> node routing -> CNI -> Pod B's network namespace
```

Il pod B riceve il pacchetto e lo elabora in base alle esigenze.

Risposta

6. Il pod B invia la risposta

Il pod B crea un pacchetto di risposta con il proprio IP come origine e l'IP del pod A come destinazione:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 8080           |                 |
| Dst: 10.0.0.56  | Dst: 52390          |                 |
+-----------------+---------------------+-----------------+
```

La CNI identifica che questo pacchetto è destinato a una rete esterna e lo passa allo stack di rete del nodo.

5. Elaborazione della rete del nodo

Il nodo determina che l'IP di destinazione (`10.0.0.56`) non appartiene alla rete locale e inoltra il pacchetto al gateway predefinito (il router locale).

4. Routing del router locale

Il router locale consulta la sua tabella di routing e determina che l'IP di destinazione (`10.0.0.56`) appartiene al CIDR del VPC (`10.0.0.0/16`). Inoltra il pacchetto al gateway che si connette a AWS.

3. Cross-Boundary Transito

Il pacchetto torna indietro attraverso la stessa connessione on-premises al VPC, superando il limite del cloud nella direzione opposta.

2. Routing del VPC

Quando il pacchetto arriva nel VPC, il sistema di routing identifica che l'IP di destinazione appartiene a una sottorete all'interno del VPC. Il pacchetto viene instradato attraverso la rete del VPC verso l'istanza EC2 che ospita il pod A.

1. Il pod A riceve la risposta

Il pacchetto arriva all'istanza EC2 e viene consegnato direttamente al pod A tramite l'ENI collegata. Poiché la CNI del VPC non utilizza reti sovrapposte per i pod nel VPC, non è necessario alcun decapsulazione aggiuntivo: il pacchetto arriva con le intestazioni originali intatte.

Questo flusso di traffico est-ovest dimostra perché i CIDR dei pod remoti devono essere configurati correttamente e instradabili da entrambe le direzioni:

- Il VPC deve disporre di percorsi per i CIDR dei pod remoti che puntano al gateway on-premises
- La rete on-premises deve disporre di percorsi per i CIDR dei pod che indirizzano il traffico verso i nodi specifici che ospitano tali pod.

## Nodi ibridi di **`riferimento nodeadm`**

La CLI Amazon EKS Hybrid Nodes (`nodeadm`) semplifica l'installazione, la configurazione, la registrazione e la disinstallazione dei componenti dei nodi ibridi. Puoi includere il `nodeadm` nelle immagini del tuo sistema operativo per automatizzare il bootstrap dei nodi ibridi, consulta the section called "Preparazione del sistema operativo" per ulteriori informazioni.

La versione del `nodeadm` per i nodi ibridi è diversa dalla versione del `nodeadm` utilizzata per avviare le istanze Amazon EC2 come nodi nei cluster Amazon EKS. Consulta la documentazione e i riferimenti per la versione del `nodeadm` appropriata. Questa pagina di documentazione riguarda la versione del `nodeadm` per i nodi ibridi.

Il codice sorgente per i nodi ibridi `nodeadm` è pubblicato nel repository. https://github.com/aws/eks-hybrid GitHub

> **Important**
>
> È necessario eseguire l'`nodeadm`operazione con un utente con root/sudo privilegi.

> **Versione node/adm richiesta per il provider di credenziali SSM.**
>
> Se si utilizza AWS Systems Manager (SSM) come provider di credenziali per i nodi ibridi, è necessario utilizzare la `nodeadm` versione `1.0.19` o successiva per le nuove installazioni e gli aggiornamenti. Le versioni precedenti di `nodeadm` contengono una chiave di firma SSM obsoleta e hanno esito negativo durante `nodeadm install` e `nodeadm upgrade` con il seguente errore di verifica della firma:
>
> ```
> "msg":"Command failed","error":"failed to install ssm installer: validating ssm-setup-cli signature: Signature Verification Error: No matching signature"
> ```
>
> Per risolvere questo errore, scarica la versione più recente di `nodeadm` prima di eseguire `nodeadm install` o. `nodeadm upgrade`

## Scarica **`nodeadm`**

La versione Hybrid Nodes di `nodeadm` è ospitata in Amazon S3 gestita da Amazon. CloudFront Per installare `nodeadm` su ciascun host on-premises puoi eseguire il seguente comando dai tuoi host on-premises.

Per host x86_64

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/nodeadm'
```

Per host ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/nodeadm'
```

Aggiungi l'autorizzazione per i file eseguibili al file binario scaricato su ciascun host.

```
chmod +x nodeadm
```

## installazione di nodeadm

Il comando `nodeadm install` viene utilizzato per installare gli artefatti e le dipendenze necessari per eseguire e unire nodi ibridi a un cluster Amazon EKS. Il comando `nodeadm install` può essere eseguito singolarmente su ciascun nodo ibrido o può essere eseguito durante le pipeline di creazione delle immagini per preinstallare le dipendenze dei nodi ibridi nelle immagini del sistema operativo.

Utilizzo

```
nodeadm install [KUBERNETES_VERSION] [flags]
```

Argomenti posizionali

(Obbligatorio) `KUBERNETES_VERSION` La versione major.minor di EKS Kubernetes da installare, ad esempio `1.32`

Flag

| Nome | Richiesto | Description |
|---|---|---|
| `-p`, `--credential-provider` | TRUE | Provider di credenziali da installare. I valori supportati sono `iam-ra` e `ssm`. Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali". |
| `-s`, `--containerd-source` | FALSE | Origine per `containerd` . `nodeadm` supporta l'installazione di `containerd` dalla distribuzione del sistema operativo, i pacchetti Docker e la possibilità di saltare l'installazione di `containerd` . |

| Nome | Richiesto | Description |
| --- | --- | --- |
| | | Valori<br><br>`distro`- Questo è il valore predefinito. `nodeadm`installer à l'ultimo `containerd` pacchetto distribuito dal sistema operativo del nodo compatibile con la versione EKS Kubernetes. `distro`non è un valore supportato per i sistemi operativi Red Hat Enterprise Linux (RHEL).<br><br>`docker`- `nodeadm` installer à l'ultimo `containerd` pacchetto creato e distribui to da Docker compatibile con la versione EKS Kubernete s. `docker`non è un valore supportato per Amazon Linux 2023.<br><br>`none`: `nodeadm` non installer à il pacchetto `containerd` . Devi installare manualmen te `containerd` prima di eseguire `nodeadm init`. |
| `-r`,<br><br>`--region` | FALSE | Specifica la AWS regione per il download di artefatti come l'agente SSM. L'impostazione predefinita è `us-west-2` . |

| Nome | Richiesto | Description |
| --- | --- | --- |
| `-t`, `--timeout` | FALSE | Durata massima del comando di installazione. L'input segue il formato della durata. Ad esempio, `1h23m`. Il timeout per il download predefinito per il comando di installazione è impostato su 20 minuti. |
| `-h`, `--help` | FALSE | Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili. |

Esempi

Installa la versione Kubernetes `1.32` con AWS Systems Manager (SSM) come fornitore di credenziali

```
nodeadm install 1.32 --credential-provider ssm
```

Installa la versione Kubernetes `1.32` con AWS Systems Manager (SSM) come fornitore di credenziali e Docker come sorgente contenuta, con un timeout di download di 20 minuti.

```
nodeadm install 1.32 --credential-provider ssm --containerd-source docker --timeout 20m
```

Installa la versione Kubernetes con IAM Roles Anywhere come provider di credenziali `1.32` AWS

```
nodeadm install 1.32 --credential-provider iam-ra
```

## `controllo della configurazione nodeadm`

Il comando `nodeadm config check` verifica la presenza di errori nella configurazione del nodo fornita. Questo comando può essere utilizzato per verificare e confermare la correttezza di un file di configurazione del nodo ibrido.

Utilizzo

```
nodeadm config check [flags]
```

Flag

| Nome | Richiesto | Description |
|---|---|---|
| `-c`, `--config-source` | TRUE | Origine della configurazione di nodeadm. Per i nodi ibridi, l'input deve seguire un URI con schema di file. |
| `-h`, `--help` | FALSE | Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili. |

Esempi

```
nodeadm config check -c file://nodeConfig.yaml
```

## `nodeadm init`

Il comando `nodeadm init` avvia e connette il nodo ibrido con il cluster Amazon EKS configurato. Consulta the section called "Configurazione del modo per attivazioni ibride SSM" o the section called "Configurazione del nodo per IAM Roles Anywhere" per i dettagli su come configurare il file `nodeConfig.yaml`.

Utilizzo

```
nodeadm init [flags]
```

Flag

| Nome | Richiesto | Description |
|---|---|---|
| `-c`, | TRUE | Origine della configurazione di `nodeadm`. Per i nodi ibridi, |

| Nome | Richiesto | Description |
| --- | --- | --- |
| `--config-source` | | l'input deve seguire un URI con schema di file. |
| `-s`,<br><br>`--skip` | FALSE | Fasi di `init` da saltare. Non è consigliabile saltare nessuna delle fasi a meno che non sia utile per risolvere un problema.<br><br>Valori<br><br>`install-validation` salta la verifica se il comando di installazione precedente è stato eseguito correttamente.<br><br>`cni-validation` salta la verifica se le porte VXLAN della CNI di Cilium o di Calico sono aperte se il firewall è abilitato sul nodo<br><br>`node-ip-validation` salta la verifica se l'IP del nodo rientra in un CIDR nelle reti dei nodi remoti |
| `-h`, `--help` | FALSE | Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili. |

Esempi

```
nodeadm init -c file://nodeConfig.yaml
```

## aggiornamento nodeadm

Il comando `nodeadm upgrade` aggiorna tutti gli artefatti installati alla versione più recente e avvia il nodo per configurare gli artefatti aggiornati e unirsi al cluster EKS su AWS. Upgrade è un comando che interrompe i carichi di lavoro in esecuzione sul nodo. Devi spostare i carichi di lavoro su un altro nodo prima di eseguire l'aggiornamento.

Utilizzo

```
nodeadm upgrade [KUBERNETES_VERSION] [flags]
```

Argomenti posizionali

(Obbligatorio) `KUBERNETES_VERSION` La versione major.minor di EKS Kubernetes da installare, ad esempio `1.32`

Flag

| Nome | Richiesto | Description |
|---|---|---|
| `-c`, `--config-source` | TRUE | Origine della configurazione di `nodeadm`. Per i nodi ibridi, l'input deve seguire un URI con schema di file. |
| `-t`, `--timeout` | FALSE | Timeout per il download degli artefatti. L'input segue il formato della durata. Ad esempio 1h23m. Il timeout per il download predefinito per il comando di aggiornamento è impostato su 10 minuti. |
| `-s`, `--skip` | FALSE | Fasi dell'aggiornamento da saltare. Non è consigliabile saltare nessuna Passaggio a meno che non sia utile per risolvere un problema. |

| Nome | Richiesto | Description |
| --- | --- | --- |
| | | Valori<br><br>`pod-validation` salta la verifica se tutti i pod sono in esecuzione sul nodo, tranne i set daemon e i pod statici.<br><br>`node-validation` salta la verifica se il nodo è stato isolato.<br><br>`init-validation` salta la verifica se il nodo è stato inizializzato correttamente prima di eseguire l'aggiorn amento.<br><br>`containerd-major-version-upgrade` impedisce gli aggiornamenti delle versioni principali di containerd durante l'aggiornamento del nodo. |
| `-h`, `--help` | FALSE | Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili. |

Esempi

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml
```

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml --timeout 20m
```

## disinstallazione di nodeadm

Il comando `nodeadm uninstall` arresta e rimuove gli artefatti che `nodeadm` installa durante `nodeadm install`, inclusi kubelet e containerd. Attenzione, il comando di disinstallazione non scarica né elimina i nodi ibridi dal cluster. Devi eseguire le operazioni di scaricamento ed eliminazione separatamente, consulta the section called "Elimina i nodi ibridi" per ulteriori informazioni. Per impostazione predefinita, `nodeadm uninstall` non proseguirà se sul nodo rimangono dei pod. Allo stesso modo, `nodeadm uninstall` non rimuove le dipendenze CNI o le dipendenze di altri componenti aggiuntivi di Kubernetes eseguiti sul cluster. Per rimuovere completamente l'installazione CNI dal tuo host, consulta le istruzioni contenute in the section called "Configurazione della CNI". Se utilizzi le attivazioni ibride AWS SSM come provider di credenziali locale, il `nodeadm uninstall` comando annulla la registrazione degli host come istanze gestite da SSM. AWS

Utilizzo

```
nodeadm uninstall [flags]
```

Flag

| Nome | Richiesto | Description |
|---|---|---|
| `-s`,<br>`--skip` | FALSE | Fasi di disinstallazione da saltare. Non è consigliabile saltare nessuna delle fasi a meno che non sia utile per risolvere un problema.<br>Valori<br>`pod-validation` salta la verifica se tutti i pod sono in esecuzione sul nodo, tranne i set daemon e i pod statici.<br>`node-validation` salta la verifica se il nodo è stato isolato. |

| Nome | Richiesto | Description |
| --- | --- | --- |
| | | `init-validation` salta la verifica se il nodo è stato inizializzato correttamente prima di eseguire la disinstal lazione. |
| `-h`,<br>`--help` | FALSE | Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili. |

| Nome | Richiesto | Description |
| --- | --- | --- |
| `-f`,<br><br>`--force` | FALSE | Forza l'eliminazione delle directory aggiuntive che potrebbero contenere file rimanenti dai componenti Kubernetes e CNI.<br><br>ATTENZIONE<br><br>Ciò eliminerà tutti i contenuti nelle directory Kubernetes e CNI predefinite (`/var/lib/cni` , `/etc/cni/net.d` , ecc.). Non utilizzare questo flag se memorizzi i tuoi dati in queste posizioni.<br><br>A partire dalla versione `v1.0.9` di nodeadm, il comando `./nodeadm uninstall --skip node-validation,pod-validation --force` non elimina più la directory `/var/lib/kubelet` . Questo perché può contenere volumi Pod e directory volume-su bpath che a volte includono il filesystem dei nodi montato.<br><br>Suggerimenti per una gestione sicura<br><br>L'eliminazione dei percorsi montati può portare alla cancellazione accidenta le dell'effettivo filesystem |

| Nome | Richiesto | Description |
| --- | --- | --- |
| | | dei nodi montati. Prima di eliminare manualmente la directory `/var/lib/kubelet` , ispeziona attentamente tutti i montaggi attivi e smonta i volumi in modo sicuro per evitare la perdita di dati. |

Esempi

```
nodeadm uninstall
```

```
nodeadm uninstall --skip node-validation,pod-validation
```

## `debug nodeadm`

Il comando `nodeadm debug` può essere utilizzato per risolvere i problemi relativi ai nodi ibridi non integri o configurati in modo errato. Verifica che i seguenti requisiti siano soddisfatti.

- Il nodo ha accesso in rete alle AWS API richieste per ottenere le credenziali,
- Il nodo è in grado di ottenere AWS le credenziali per il ruolo IAM di Hybrid Nodes configurato,
- Il nodo ha accesso di rete all'endpoint dell'API EKS Kubernetes e ha la validità del certificato endpoint dell'API EKS Kubernetes.
- Il nodo è in grado di autenticarsi con il cluster EKS, la sua identità nel cluster è valida e ha accesso al cluster EKS tramite il VPC configurato per il cluster EKS.

Se vengono rilevati errori, l'output del comando suggerisce le procedure per la risoluzione dei problemi. Alcuni passaggi di conferma mostrano le procedure secondarie. Se queste non funzionano, l'output viene mostrato in una sezione stderr sotto l'errore di convalida.

Utilizzo

```
nodeadm debug [flags]
```

Flag

| Nome | Richiesto | Description |
|---|---|---|
| `-c`, `--config-source` | TRUE | Origine della configurazione di `nodeadm`. Per i nodi ibridi, l'input deve seguire un URI con schema di file. |
| `--no-color` | FALSE | Disattiva l'output a colori. Utile per l'automazione. |
| `-h`, `--help` | FALSE | Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili. |

Esempi

```
nodeadm debug -c file://nodeConfig.yaml
```

## Posizioni dei file nodeadm

installazione di nodeadm

Durante l'esecuzione di `nodeadm install`, vengono configurati i seguenti file e posizioni dei file.

| Artifact | Path |
|---|---|
| CLI IAM Roles Anywhere | /usr/local/bin/aws_signing_helper |
| Binario Kubelet | //kubelet usr/bin |
| Binario Kubectl | usr/local/bin/kubectl |
| Provider di credenziali ECR | //image-credential- -fornitore-credenziali etc/eks provider/ecr |
| AWS Autenticatore IAM | //usr/local-iam-authenticator bin/aws |

| Artifact | Path |
| --- | --- |
| CLI di configurazione SSM | opt/ssm//ssm-setup-cli |
| Agente SSM | Su Ubuntu -/-ssm- /amazon-ssm-agent snap/amazon agent/current<br><br>Su RHEL e AL2023 -/usr/bin/amazon-ssm-agent |
| Containerd | Su Ubuntu e AL2023 -usr/bin//containerd<br><br>Su RHEL -/bin/containerd |
| Iptables | Su Ubuntu e AL2023 -//iptables usr/sbin<br><br>Su RHEL -/sbin/iptables |
| Plug-in CNI | //bin opt/cni |
| tracciatore di artefatti installati | //localizzatore opt/nodeadm |

nodeadm init

Durante l'esecuzione di `nodeadm init`, vengono configurati i file e le posizioni seguenti dei file.

| Nome | Path |
| --- | --- |
| Kubelet kubeconfig | /var/lib/kubelet/kubeconfig |
| Configurazione di Kubelet | //.json etc/kubernetes kubelet/config |
| Unità Kubelet systemd | //.servizio etc/systemd system/kubelet |
| Configurazione del provider di credenziali di immagine | /etc/eksprovider/config/image-credential- .json |
| File env Kubelet | /etc/eks/kubelet/environment |
| Certificati Kubelet | etc/kubernetes//.crt pki/ca |

| Nome | Path |
| --- | --- |
| Configurazione di Containerd | //config.toml etc/containerd |
| Configurazione dei moduli del kernel Container d | etc/modules/-carica. d/containerd.conf |
| AWS file di configurazione | /etc/aws/hybrid/config |
| AWS file delle credenziali (se abilita il file delle credenziali) | /eks-hybrid/. aws/credentials |
| AWS unità di sistema di supporto alla firma | /etc/systemd/.servizio system/aws_signing _helper_update |
| File conf sysctl | /etc/sysctl. d/99-nodeadm.conf |
| Ca-certificates | //-certificates.crt etc/ssl certs/ca |
| File chiave Gpg | etc/apt/keyrings/docker/.asc |
| File di origine del repository Docker | //sources.list. etc/apt d/docker.lista |

## Configurazione del modo per attivazioni ibride SSM

Di seguito è riportato un esempio `nodeConfig.yaml` di utilizzo delle attivazioni ibride AWS SSM per le credenziali dei nodi ibridi.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

## Configurazione del nodo per IAM Roles Anywhere

Di seguito è riportato un esempio di `nodeConfig.yaml` credenziali AWS IAM Roles Anywhere for hybrid nodes.

Quando si utilizza AWS IAM Roles Anywhere come provider di credenziali locale, le `nodeName` credenziali utilizzate nella `nodeadm` configurazione devono essere in linea con le autorizzazioni previste per il ruolo IAM di Hybrid Nodes. Ad esempio, se le autorizzazioni per il ruolo IAM di Hybrid Nodes consentono a AWS IAM Roles Anywhere di assumere il ruolo solo quando il nome della sessione del ruolo è uguale al CN del certificato host, allora il CN `nodeName` nella `nodeadm` configurazione deve essere uguale al CN dei certificati. Il `nodeName` che usi non può contenere più di 64 caratteri. Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:              # Name of the EKS cluster
    region:            # AWS Region where the EKS cluster resides
  hybrid:
    iamRolesAnywhere:
      nodeName:        # Name of the node
      trustAnchorArn:  # ARN of the IAM Roles Anywhere trust anchor
      profileArn:      # ARN of the IAM Roles Anywhere profile
      roleArn:         # ARN of the Hybrid Nodes IAM role
      certificatePath: # Path to the certificate file to authenticate with the IAM
 Roles Anywhere trust anchor
      privateKeyPath:  # Path to the private key file for the certificate
```

## Configurazione del nodo per personalizzare kubelet (Facoltativo)

Puoi passare la configurazione e i flag di kubelet nella tua configurazione `nodeadm`. Vedi l'esempio seguente per come aggiungere un'etichetta di nodo aggiuntiva `abc.example.com/test-label` e impostare la configurazione di kubelet su 30 secondi. `shutdownGracePeriod` Per ulteriori informazioni sulle opzioni di configurazione di Kubelet, consulta il riferimento alla configurazione di Kubelet (v1beta1) nella documentazione di Kubernetes. Per ulteriori informazioni sui flag della riga di comando kubelet, consulta il riferimento alla CLI di kubelet nella documentazione di Kubernetes. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

```
apiVersion: node.eks.aws/v1alpha1
```

```
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  kubelet:
    config:           # Map of kubelet config and values
       shutdownGracePeriod: 30s
    flags:            # List of kubelet flags
       - --node-labels=abc.example.com/test-label=true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

## Configurazione del nodo per personalizzare containerd (Facoltativo)

Puoi passare la configurazione containerd personalizzata nella tua configurazione `nodeadm`. La configurazione containerd per `nodeadm` accetta TOML in linea. Vedi l'esempio seguente su come configurare containerd per disabilitare l'eliminazione dei livelli di immagine decompressi nel content store containerd.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri".containerd]
       discard_unpacked_layers = false
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

> **Note**
>
> Le versioni 1.x e 2.x di Containerd utilizzano diversi formati di configurazione. Containerd 1.x utilizza la versione di configurazione 2, mentre containerd 2.x utilizza la versione di configurazione 3. Sebbene containerd 2.x rimanga retrocompatibile con la versione 2 di

configurazione, si consiglia la versione 3 di config per prestazioni ottimali. Controlla la tua versione di containerd o consulta i log di installazione. `containerd --version` nodeadm Per maggiori dettagli sul controllo delle versioni di configurazione, vedi https://containerd.io/releases/

Puoi anche utilizzare la configurazione Containerd per abilitare il supporto di SELinux. Con SELinux abilitato su containerd, assicurati che i pod pianificati sul nodo abbiano il SecurityContext corretto e siano abilitati. LinuxOptions Ulteriori informazioni sulla configurazione di un contesto di sicurezza sono disponibili in Kubernetes documentation.

**Note**

Red Hat Enterprise Linux (RHEL) 8 e RHEL 9 hanno SELinux abilitato di default e impostato su strict sull'host. Per impostazione predefinita, SELinux è abilitato e impostato sulla modalità di autorizzazione in Amazon Linux 2023. Quando SELinux è impostato sulla modalità di autorizzazione sull'host, abilitarlo su containerd non bloccherà le richieste ma le registrerà in base alla configurazione SELinux sull'host.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri"]
       enable_selinux = true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

## Risoluzione dei problemi relativi ai nodi ibridi

Questo argomento illustra alcuni errori comuni che si potrebbero verificare durante l'utilizzo di Amazon EKS Hybrid Nodes, e il modo in cui ripararli. Per altre informazioni sulla risoluzione dei

problemi, consulta Risoluzione dei problemi il tag Knowledge Center per Amazon EKS su AWS re:POST. Se non riesci a risolvere il problema, contatta l'assistenza. AWS

⚠ Versione node/adm richiesta per il provider di credenziali SSM

Se si utilizza AWS Systems Manager (SSM) come provider di credenziali per i nodi ibridi, è necessario utilizzare la `nodeadm` versione `1.0.19` o successiva per le nuove installazioni e gli aggiornamenti. Le versioni precedenti di `nodeadm` contengono una chiave di firma SSM obsoleta e hanno esito negativo durante `nodeadm install` e `nodeadm upgrade` con il seguente errore di verifica della firma:

```
"msg":"Command failed","error":"failed to install ssm installer: validating ssm-setup-cli signature: Signature Verification Error: No matching signature"
```

Per risolvere questo errore, scarica la versione più recente di `nodeadm` prima di eseguire `nodeadm install` o. `nodeadm upgrade`

Risoluzione dei problemi dei nodi con **`nodeadm debug`**. Puoi eseguire il comando `nodeadm debug` dai nodi ibridi per verificare che i requisiti di rete e le credenziali siano soddisfatti. Per ulteriori informazioni sul comando `nodeadm debug`, consulta the section called "nodeadm dei nodi ibridi".

Rileva i problemi con i tuoi nodi ibridi con le informazioni sul cluster. Le informazioni sul cluster Amazon EKS includono controlli approfonditi che rilevano problemi comuni con la configurazione dei nodi ibridi EKS nel cluster. Puoi visualizzare i risultati di tutti i controlli approfonditi da Console di gestione AWS, AWS CLI e AWS SDK. Per ulteriori informazioni sulle informazioni sul cluster, consulta the section called "Approfondimenti sui cluster".

## Risoluzione dei problemi di installazione dei nodi ibridi

I seguenti argomenti di risoluzione dei problemi riguardano l'installazione delle dipendenze dei nodi ibridi sugli host con il comando `nodeadm install`.

### Comando **`nodeadm`** non riuscito per **`must run as root`**

Il comando `nodeadm install` deve essere eseguito con un utente che dispone di root o privilegi `sudo` sull'host. Se esegui `nodeadm install` con un utente che non dispone di root o privilegi `sudo`, nell'output `nodeadm` verrà visualizzato il seguente errore.

```
"msg":"Command failed","error":"must run as root"
```

Impossibile connettersi alle dipendenze

Il comando `nodeadm install` installa le dipendenze richieste per i nodi ibridi. Le dipendenze dei nodi ibridi includonocontainerd, `kubeletkubectl`, e i componenti AWS SSM o AWS IAM Roles Anywhere. Devi avere accesso da dove stai eseguendo `nodeadm install` per scaricare queste dipendenze. Per ulteriori informazioni sull'elenco delle posizioni a cui devi poter accedere, consulta the section called "Preparazione della rete". Se non disponi dell'accesso, nell'output `nodeadm install` verranno visualizzati errori simili ai seguenti.

```
"msg":"Command failed","error":"failed reading file from url: ...: max retries achieved
 for http request"
```

Impossibile aggiornare il gestore di pacchetti

Il comando `nodeadm install` esegue `apt update`, `yum update` o `dnf update` prima dell'installazione delle dipendenze dei nodi ibridi. Se questo passaggio non ha esito positivo, potresti visualizzare errori simili ai seguenti. Per rimediare, puoi eseguire `apt update`, `yum update` o `dnf update` prima di eseguire `nodeadm install` oppure puoi provare a rieseguire `nodeadm install`.

```
failed to run update using package manager
```

Scadenza di un timeout o di un contesto superata

Durante l'esecuzione di `nodeadm install`, se riscontri problemi in varie fasi del processo di installazione con errori che indicano che è stata superata la scadenza di un timeout o di un contesto, è possibile che la connessione sia lenta e impedisca l'installazione delle dipendenze dei nodi ibridi prima del raggiungimento dei timeout. Per risolvere questi problemi, puoi provare a utilizzare il flag `--timeout` in `nodeadm` per estendere la durata dei timeout per il download delle dipendenze.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --timeout 20m0s
```

## Risoluzione dei problemi di connessione dei nodi ibridi

Gli argomenti di risoluzione dei problemi in questa sezione riguardano il processo di connessione dei nodi ibridi ai cluster EKS con il comando `nodeadm init`.

Errori operativi o schema non supportato

Durante l'esecuzione di `nodeadm init`, se vedi errori relativi a `operation error` o `unsupported scheme`, controlla `nodeConfig.yaml` per assicurarti che sia formattato e passato correttamente a `nodeadm`. Per ulteriori informazioni sul formato e le opzioni per `nodeConfig.yaml`, consulta the section called "nodeadm dei nodi ibridi".

```
"msg":"Command failed","error":"operation error ec2imds: GetRegion, request canceled,
 context deadline exceeded"
```

Al ruolo IAM dei nodi ibridi mancano le autorizzazioni per l'azione **`eks:DescribeCluster`**

Durante l'esecuzione`nodeadm init`, `nodeadm` tenta di raccogliere informazioni sul cluster EKS richiamando l'azione EKS`DescribeCluster`. Se il ruolo IAM di Hybrid Nodes non dispone dell'autorizzazione per l'`eks:DescribeCluster`azione, devi passare l'endpoint dell'API Kubernetes, il bundle CA del cluster e il CIDR IPv4 del servizio nella configurazione del nodo a cui passi durante l'esecuzione. `nodeadm` `nodeadm init` Per ulteriori informazioni sulle autorizzazioni richieste per il ruolo IAM dei nodi ibridi, consulta the section called "Preparazione delle credenziali".

```
"msg":"Command failed","error":"operation error EKS: DescribeCluster, https response
 error StatusCode: 403 ... AccessDeniedException"
```

Al ruolo IAM dei nodi ibridi mancano le autorizzazioni per l'azione **`eks:ListAccessEntries`**

Durante l'esecuzione`nodeadm init`, `nodeadm` tenta di convalidare se il cluster EKS ha una voce di accesso di tipo `HYBRID_LINUX` associata al ruolo IAM di Hybrid Nodes chiamando l'azione EKS. `ListAccessEntries` Se il ruolo IAM di Hybrid Nodes non dispone dell'autorizzazione per l'`eks:ListAccessEntries`azione, è necessario passare il `--skip cluster-access-validation` flag quando si esegue il `nodeadm init` comando. Per ulteriori informazioni sulle autorizzazioni richieste per il ruolo IAM dei nodi ibridi, consulta the section called "Preparazione delle credenziali".

```
"msg":"Command failed","error":"operation error EKS: ListAccessEntries, https response
 error StatusCode: 403 ... AccessDeniedException"
```

IP del nodo non presente nella rete di nodi remoti CIDR

Durante l'esecuzione di `nodeadm init`, potresti riscontrare un errore se l'indirizzo IP del nodo non rientra nei CIDR della rete del nodo remoto specificati. L'aspetto dell'errore sarà simile all'esempio seguente:

```
node IP 10.18.0.1 is not in any of the remote network CIDR blocks [10.0.0.0/16
 192.168.0.0/16]
```

Questo esempio mostra un nodo con IP 10.18.0.1 che tenta di unirsi a un cluster con una rete remota CIDRS 10.0.0. 0/16 e 192.168.0. 0/16. L'errore si verifica perché 10.18.0.1 non rientra in nessuno degli intervalli.

Conferma di aver configurato correttamente i tuoi indirizzi IP `RemoteNodeNetworks` per includere tutti gli indirizzi IP del nodo. Per ulteriori informazioni sulla configurazione delle reti, consulta the section called "Preparazione della rete".

- Esegui il comando seguente nella regione in cui si trova il cluster per verificare le configurazioni `RemoteNodeNetwork`. Verifica che i blocchi CIDR elencati nell'output includano l'intervallo IP del nodo e corrispondano ai blocchi CIDR elencati nel messaggio di errore. Se non corrispondono, conferma che il nome e la regione del cluster `nodeConfig.yaml` corrispondano al cluster desiderato.

```
aws eks describe-cluster --name CLUSTER_NAME --region REGION_NAME --query
 cluster.remoteNetworkConfig.remoteNodeNetworks
```

- Verifica di lavorare con il nodo desiderato:
  - Conferma di essere sul nodo corretto controllando che il nome host e l'indirizzo IP corrispondano a quelli che intendi registrare nel cluster.
  - Verifica che questo nodo si trovi nella rete on-premises corretta (quella il cui intervallo CIDR è stato registrato come `RemoteNodeNetwork` durante la configurazione del cluster).

Se l'IP del nodo non è ancora quello previsto, controlla quanto segue:

- Se utilizzi IAM Roles Anywhere, `kubelet` esegue una ricerca DNS su IAM Roles Anywhere `nodeName` e utilizza un IP associato al nome del nodo, se disponibile. Se gestisci gli ingressi DNS per i tuoi nodi, conferma che questi punti di ingresso agli IP all'interno dei CIDR della rete del nodo remoto.
- Se il tuo nodo ha più interfacce di rete, `kubelet` potrebbe selezionare un'interfaccia con un indirizzo IP esterno ai CIDR della rete del nodo remoto come impostazione predefinita. Per utilizzare un'interfaccia diversa, specifica il relativo indirizzo IP utilizzando il flag `--node-ip` `kubelet` nel tuo `nodeConfig.yaml`. Per ulteriori informazioni, consulta the section called

"nodeadm dei nodi ibridi". Puoi visualizzare le interfacce di rete del tuo nodo e i relativi indirizzi IP eseguendo il seguente comando sul tuo nodo:

```
ip addr show
```

I nodi ibridi non vengono visualizzati nel cluster EKS

Se l'hai eseguito `nodeadm init` ed è stato completato ma i nodi ibridi non sono visualizzati nel cluster, potrebbero esserci problemi con la connessione di rete tra i nodi ibridi e il piano di controllo EKS, potresti non avere le autorizzazioni richieste per il gruppo di sicurezza configurate o potresti non avere la mappatura richiesta del ruolo IAM di Hybrid Nodes su Kubernetes Role-Based Access Control (RBAC). Puoi avviare il processo di debug controllando lo stato di `kubelet` e i log `kubelet` con i seguenti comandi. Esegui i seguenti comandi dai nodi ibridi che non sono riusciti a entrare a far parte del cluster.

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Impossibile comunicare con il cluster

Se il nodo ibrido non è in grado di comunicare con il piano di controllo del cluster, potresti visualizzare log simili ai seguenti.

```
"Failed to ensure lease exists, will retry" err="Get ..."
```

```
"Unable to register node with API server" err="Post ..."
```

```
Failed to contact API server when waiting for CSINode publishing ... dial tcp <ip
 address>: i/o timeout
```

Se visualizzi questi messaggi, controlla quanto segue per assicurarti che soddisfi i requisiti dei nodi ibridi descritti in the section called "Preparazione della rete".

- Verifica che il VPC passato al cluster EKS disponga di percorsi verso il tuo Transit Gateway (TGW) o Gateway privato virtuale (VGW) per il nodo on-premises e, facoltativamente, i pod CIDR.

- Conferma di disporre di un gruppo di sicurezza aggiuntivo per il cluster EKS che disponga di regole in entrata per i CIDR del nodo on-premises e, facoltativamente, per i CIDR del pod.
- Verifica che il router on-premises sia configurato per consentire il traffico da e verso il piano di controllo EKS.

Non autorizzato

Se il nodo ibrido è in grado di comunicare con il piano di controllo EKS ma non è in grado di registrarsi, potresti visualizzare log simili ai seguenti. Nota che la differenza principale nei messaggi di log riportati di seguito è l'errore `Unauthorized`. Ciò indica che il nodo non è stato in grado di eseguire le proprie attività perché non dispone delle autorizzazioni richieste.

```
"Failed to ensure lease exists, will retry" err="Unauthorized"
```

```
"Unable to register node with API server" err="Unauthorized"
```

```
Failed to contact API server when waiting for CSINode publishing: Unauthorized
```

Se visualizzi questi messaggi, controlla quanto segue per assicurarti che soddisfi i dettagli dei requisiti dei nodi in the section called "Preparazione delle credenziali" e the section called "Preparazione dell'accesso ai cluster".

- Verifica che l'identità dei nodi ibridi corrisponda al ruolo IAM dei nodi ibridi previsto. Questo può essere fatto eseguendo `sudo aws sts get-caller-identity` dai tuoi nodi ibridi.
- Verifica che il ruolo IAM dei nodi ibridi disponga delle autorizzazioni richieste.
- Conferma che nel tuo cluster hai una voce di accesso EKS per il tuo ruolo IAM di Hybrid Nodes o conferma di avere una voce per il tuo `aws-auth` ConfigMap ruolo IAM di Hybrid Nodes. Se utilizzi le voci di accesso EKS, conferma che la voce di accesso per il ruolo IAM dei nodi ibridi abbia il tipo di accesso `HYBRID_LINUX`. Se stai utilizzando il `aws-auth` ConfigMap, conferma che la tua voce per il ruolo IAM di Hybrid Nodes soddisfi i requisiti e la formattazione descritti inthe section called "Preparazione dell'accesso ai cluster".

I nodi ibridi sono registrati nel cluster EKS ma mostrano lo stato **`Non pronti`**

Se i nodi ibridi sono registrati correttamente nel cluster EKS, ma mostrano lo stato `Not Ready`, la prima cosa da verificare è lo stato dell'interfaccia CNI (Container Networking Interface). Se non

hai installato una CNI, si prevede che i nodi ibridi abbiano lo status `Not Ready`. Quando una CNI è installata e funziona correttamente, i nodi vengono aggiornati allo stato `Ready`. Se hai tentato di installare una CNI ma non funziona correttamente, consulta the section called "Risoluzione dei problemi CNI dei nodi ibridi" su questa pagina.

Le richieste di firma certificati (CSR) sono bloccate su In sospeso

Dopo aver collegato i nodi ibridi al cluster EKS, se noti che ci sono CSR in sospeso per i nodi ibridi, significa che i nodi ibridi non soddisfano i requisiti per l'approvazione automatica. Le CSR per i nodi ibridi vengono approvate ed emesse automaticamente se le CSR per i nodi ibridi sono state create da un nodo con etichetta `eks.amazonaws.com/compute-type: hybrid` e la CSR ha i seguenti Nomi alternativi oggetto (SAN): almeno una SAN DNS è uguale al nome del nodo e le SAN IP appartengono ai CIDR della rete del nodo remoto.

Il profilo ibrido esiste già

Se hai modificato la configurazione `nodeadm` e tenti di registrare nuovamente il nodo con la nuova configurazione, potresti visualizzare un errore che indica che il profilo ibrido esiste già ma il suo contenuto è cambiato. Invece di eseguire l'operazione `nodeadm init` tra le modifiche alla configurazione, esegui `nodeadm uninstall` seguito da `nodeadm install` e `nodeadm init`. Ciò garantisce una corretta pulizia con le modifiche alla configurazione.

```
"msg":"Command failed","error":"hybrid profile already exists at /etc/aws/hybrid/config
 but its contents do not align with the expected configuration"
```

Il nodo ibrido non è riuscito a risolvere l'API privata

Dopo l'esecuzione di `nodeadm init`, se visualizzi un errore nei log `kubelet` che indica che non è stato possibile contattare il server dell'API EKS Kubernetes perché si è verificato `no such host`, potresti dover modificare la voce DNS per l'endpoint dell'API EKS Kubernetes nella rete on-premises o a livello di host. Vedi Inoltro di richieste DNS in entrata al tuo VPC nella documentazione di Route53. AWS

```
Failed to contact API server when waiting for CSINode publishing: Get ... no such host
```

Impossibile visualizzare i nodi ibridi nella console EKS

Se hai registrato i tuoi nodi ibridi ma non riesci a visualizzarli nella console EKS, controlla le autorizzazioni dell'IAM principale che stai utilizzando per visualizzare la console. L'IAM principale che

stai utilizzando deve disporre di autorizzazioni IAM e Kubernetes minime specifiche per visualizzare le risorse nella console. Per ulteriori informazioni, consulta the section called "Accesso alle risorse di cluster".

## Risoluzione dei problemi di esecuzione dei nodi ibridi

Se i nodi ibridi che sono stati registrati nel cluster EKS avevano lo stato `Ready` e poi sono passati allo stato `Not Ready`, esistono molti problemi che potrebbero aver contribuito a tale stato di non integrità, ad esempio la mancanza di risorse sufficienti per la CPU, la memoria o lo spazio su disco disponibile del nodo, oppure la disconnessione del nodo dal piano di controllo EKS. Puoi utilizzare i passaggi seguenti per risolvere i problemi dei tuoi nodi e, se non riesci a risolvere il problema, contatta l'assistenza. AWS

Esegui `nodeadm debug` dai nodi ibridi per verificare che i requisiti di rete e le credenziali siano soddisfatti. Per ulteriori informazioni sul comando `nodeadm debug`, consulta the section called "nodeadm dei nodi ibridi".

Ottieni lo stato del nodo

```
kubectl get nodes -o wide
```

Controlla le condizioni e gli eventi del nodo

```
kubectl describe node NODE_NAME
```

Ottieni lo stato del pod

```
kubectl get pods -A -o wide
```

Controlla le condizioni e gli eventi del pod

```
kubectl describe pod POD_NAME
```

Controlla i log del pod

```
kubectl logs POD_NAME
```

Controlla i log di **`kubelet`**

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Probe liveness del pod non funziona o i webhook non funzionano

Se le applicazioni, i componenti aggiuntivi o i webhook in esecuzione sui nodi ibridi non si avviano correttamente, potresti riscontrare problemi di rete che bloccano la comunicazione con i pod. Affinché il piano di controllo EKS contatti i webhook in esecuzione su nodi ibridi, è necessario configurare il cluster EKS con una rete pod remota e inserire i percorsi per il POD CIDR locale nella tabella di routing VPC con la destinazione come Transit Gateway (TGW), virtual private gateway (VGW) o altro gateway utilizzato per connettere il VPC alla rete locale. Per ulteriori informazioni sui requisiti di rete per i nodi ibridi, consulta the section called "Preparazione della rete". Devi inoltre consentire a questo traffico di entrare nel firewall on-premises e assicurarti che il router sia in grado di instradare correttamente i pod. Consulta the section called "Configurazione di webhook" per ulteriori informazioni sui requisiti per l'esecuzione di webhook su nodi ibridi.

Di seguito è riportato un messaggio di log del pod comune per questo scenario, dove l'indirizzo IP è l'IP del cluster per il servizio Kubernetes.

```
dial tcp <ip-address>:443: connect: no route to host
```

**`kubectl logs`**o comandi non funzionanti (comandi API) **`kubectl exec kubelet`**

Se`kubectl attach`, `kubectl cp` `kubectl exec`, `kubectl logs`, e `kubectl port-forward` i comandi scadono mentre altri `kubectl` comandi hanno esito positivo, il problema è probabilmente correlato alla configurazione della rete remota. Questi comandi si connettono tramite il cluster all'endpoint `kubelet` sul nodo. Per ulteriori informazioni, consulta the section called "Endpoint `kubelet`".

Verifica che gli IP dei nodi e gli IP dei pod rientrino nei CIDR della rete del nodo remoto e della rete del pod remoto configurati per il cluster. Usa i comandi seguenti per esaminare le assegnazioni IP.

```
kubectl get nodes -o wide
```

```
kubectl get pods -A -o wide
```

Confronta questi IP con i CIDR di rete remota configurati per garantire un routing corretto. Per i requisiti relativi alla configurazione della rete, consulta the section called "Preparazione della rete".

## Risoluzione dei problemi CNI dei nodi ibridi

Se riscontri problemi con l'avvio iniziale di Cilium o Calico con nodi ibridi, il più delle volte ciò è dovuto a problemi di rete tra i nodi ibridi o i pod CNI in esecuzione sui nodi ibridi e il piano di controllo EKS. Assicurati che il tuo ambiente soddisfi i requisiti di Preparazione della rete per i nodi ibridi. È utile suddividere il problema in parti.

Configurazione del cluster EKS

- Le RemotePodNetwork configurazioni RemoteNodeNetwork e sono corrette?

Configurazione VPC

- Esistono percorsi per l' RemoteNodeNetwork e RemotePodNetwork nella tabella di routing del VPC con destinazione Transit Gateway o Virtual Private Gateway?

Configurazione del gruppo di sicurezza

- Esistono regole in entrata e in uscita per la destinazione finale? RemoteNodeNetwork RemotePodNetwork

On-premises rete

- Esistono percorsi e accessi da e verso il piano di controllo EKS e da e verso i nodi ibridi e i pod in esecuzione sui nodi ibridi?

Configurazione CNI

- Se si utilizza una rete overlay, la configurazione del pool IP per il CNI corrisponde a quella RemotePodNetwork configurata per il cluster EKS se si utilizzano webhook?

Il nodo ibrido ha lo stato **`Ready`** senza una CNI installata

Se i nodi ibridi mostrano lo stato `Ready`, ma non è stata installata una CNI sul cluster, è possibile che sui nodi ibridi siano presenti vecchi artefatti CNI. Per impostazione predefinita, quando si disinstallano Cilium e Calico con strumenti come Helm, le risorse su disco non vengono rimosse dalle macchine

fisiche o virtuali. Inoltre, le Definizioni delle risorse personalizzate (CRD) per questi CNI potrebbero essere ancora presenti nel cluster da una vecchia installazione. Per ulteriori informazioni, consulta le sezioni Eliminazione Cilium ed Eliminazione Calico di the section called "Configurazione della CNI".

### Risoluzione dei problemi di Cilium

Se riscontri problemi nell'esecuzione di Cilium su nodi ibridi, consulta i passaggi per la risoluzione dei problemi nella documentazione di Cilium. Le sezioni seguenti trattano i problemi che potrebbero riguardare esclusivamente la distribuzione di Cilium sui nodi ibridi.

#### Cilium non si avvia

Se gli agenti Cilium eseguiti su ciascun nodo ibrido non si avviano, controlla se sono presenti errori nei log dei pod degli agenti Cilium. L'agente Cilium richiede la connettività all'endpoint dell'API EKS Kubernetes per iniziare. L'avvio dell'agente Cilium avrà esito negativo se questa connettività non è configurata correttamente. In questo caso, nei log del pod dell'agente Cilium verranno visualizzati messaggi di registro simili ai seguenti.

```
msg="Unable to contact k8s api-server"
level=fatal msg="failed to start: Get \"https://<k8s-cluster-ip>:443/api/v1/namespaces/
kube-system\": dial tcp <k8s-cluster-ip>:443: i/o timeout"
```

L'agente Cilium viene eseguito sulla rete host. Il cluster EKS deve essere configurato con `RemoteNodeNetwork` per la connettività Cilium. Verifica di avere un gruppo di sicurezza aggiuntivo per il tuo cluster EKS con una regola in entrata per `RemoteNodeNetwork`, di disporre di percorsi nel VPC per `RemoteNodeNetwork` e che la tua rete on-premises sia configurata correttamente per consentire la connettività al piano di controllo EKS.

Se l'operatore Cilium è in esecuzione e alcuni, ma non tutti, dei tuoi agenti Cilium sono in esecuzione, conferma di avere IP pod disponibili da allocare per tutti i nodi del cluster. Configura la dimensione dei pod CIDR allocabili quando utilizzi il pool di cluster IPAM con `clusterPoolIPv4PodCIDRList` nella configurazione Cilium. La dimensione CIDR per nodo è configurata con l'impostazione `clusterPoolIPv4MaskSize` nella configurazione di Cilium. Per ulteriori informazioni, consulta Expanding the cluster pool nella documentazione di Cilium.

#### Cilium BGP non funziona

Se utilizzi il piano di controllo di Cilium BGP per pubblicizzare i tuoi pod o gli indirizzi di servizio sulla tua rete on-premises, puoi utilizzare i seguenti comandi Cilium CLI per verificare se BGP sta

pubblicizzando i percorsi verso le tue risorse. Per i passaggi per installare la CLI di Cilium, consulta Install the Cilium CLI nella documentazione di Cilium.

Se BGP funziona correttamente, dovresti vedere i tuoi nodi ibridi con Session State nell'output. `established` Potresti dover collaborare con il team di rete per identificare i valori corretti per Local AS, Peer AS e Peer Address del proprio ambiente.

```
cilium bgp peers
```

```
cilium bgp routes
```

Se utilizzi Cilium BGP per pubblicizzare gli IP dei Servizi di tipo `LoadBalancer`, devi avere la stessa etichetta sia su `CiliumLoadBalancerIPPool` che su Servizio, che deve essere utilizzata nel selettore di `CiliumBGPAdvertisement`. Di seguito è riportato un esempio. Nota, se utilizzi Cilium BGP per pubblicizzare gli IP dei Services in base al tipo LoadBalancer, le route BGP potrebbero essere interrotte durante il riavvio dell'agente Cilium. Per ulteriori informazioni, consulta Failure Scenarios nella documentazione di Cilium.

Servizio

```
kind: Service
apiVersion: v1
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
  - port: 3000
    targetPort: http-server
  selector:
    app: guestbook
  type: LoadBalancer
```

CiliumLoadBalancerIPPool

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: guestbook-pool
```

```
  labels:
    app: guestbook
spec:
  blocks:
  - cidr: <CIDR to advertise>
  serviceSelector:
    matchExpressions:
      - { key: app, operator: In, values: [ guestbook ] }
```

CiliumBGPAdvertisement

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements-guestbook
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - ExternalIP
          - LoadBalancerIP
      selector:
        matchExpressions:
          - { key: app, operator: In, values: [ guestbook ] }
```

Risoluzione dei problemi di Calico

Se riscontri problemi nell'esecuzione di Calico su nodi ibridi, consulta troubleshooting steps nella documentazione di Calico. Le sezioni seguenti trattano i problemi che potrebbero riguardare esclusivamente la distribuzione di Calico sui nodi ibridi.

La tabella seguente riassume i componenti di Calico e se funzionano di default sul nodo o sulla rete di pod. Se hai configurato Calico per utilizzare NAT per il traffico dei pod in uscita, la rete on-premises deve essere configurata per indirizzare il traffico verso il nodo on-premises CIDR e le tabelle di routing VPC devono essere configurate con una route per il nodo on-premises CIDR con il gateway di transito (TGW) o il gateway privato virtuale (VGW) come destinazione. Se non hai configurato Calico per utilizzare il NAT per il traffico dei pod in uscita, la rete on-premises deve essere configurata per indirizzare il traffico verso il pod on-premises CIDR e le tabelle di routing VPC devono essere

configurate con una route per il pod on-premises CIDR con il gateway di transito (TGW) o il gateway privato virtuale (VGW) come destinazione.

| Componente | Rete |
| --- | --- |
| Server API Calico | Nodo |
| Controller per Kubernetes di Calico | Pod |
| Agente del nodo di Calico | Nodo |
| Calico `typha` | Nodo |
| Driver del nodo CSI di Calico | Pod |
| Operatore Calico | Nodo |

Le risorse di Calico sono pianificate o in esecuzione su nodi isolati

Le risorse Calico che non vengono eseguite come a DaemonSet hanno tolleranze flessibili per impostazione predefinita che consentono di pianificarle su nodi isolati che non sono pronti per la pianificazione o l'esecuzione dei pod. È possibile inasprire le tolleranze per le risorse diverse da DaemonSet Calico modificando l'installazione dell'operatore per includere quanto segue.

```
installation:
  ...
  controlPlaneTolerations:
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  calicoKubeControllersDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
```

```
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
  typhaDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
```

## Risoluzione dei problemi relativi alle credenziali

Sia per le attivazioni ibride AWS SSM che per AWS IAM Roles Anywhere, puoi convalidare che le credenziali per il ruolo IAM di Hybrid Nodes siano configurate correttamente sui tuoi nodi ibridi eseguendo il seguente comando dai tuoi nodi ibridi. Conferma che il nome del nodo e il nome del ruolo IAM dei nodi ibridi sono quelli che ti aspetti.

```
sudo aws sts get-caller-identity
```

```
{
    "UserId": "ABCDEFGHIJKLM12345678910:<node-name>",
    "Account": "<aws-account-id>",
    "Arn": "arn:aws: sts::<aws-account-id>:assumed-role/<hybrid-nodes-iam-role/<node-
name>"
}
```

### Risoluzione dei problemi di AWS Systems Manager (SSM)

Se utilizzi attivazioni ibride AWS SSM per le credenziali dei tuoi nodi ibridi, tieni presente le seguenti directory e artefatti SSM che vengono installati sui tuoi nodi ibridi da. `nodeadm` Per ulteriori

informazioni sull'agente SSM, vedere Lavorare con l'agente SSM nella Guida per l'utente di Systems Manager. AWS

| Description | Location (Ubicazione) |
| --- | --- |
| Agente SSM | Ubuntu - `/snap/amazon-ssm-agent/current/amazon-ssm-agent` RHEL & AL2023 - `/usr/bin/amazon-ssm-agent` |
| Log di SSM Agent | `/var/log/amazon/ssm` |
| AWS credenziali | `/root/.aws/credentials` |
| CLI di configurazione SSM | `/opt/ssm/ssm-setup-cli` |

Riavvio dell'agente SSM

Alcuni problemi possono essere risolti riavviando l'agente SSM. Puoi utilizzare i comandi seguenti per riavviarlo.

AL2023 e altri sistemi operativi

```
systemctl restart amazon-ssm-agent
```

Ubuntu

```
systemctl restart snap.amazon-ssm-agent.amazon-ssm-agent
```

Verifica la connettività agli endpoint SSM

Conferma di poter effettuare la connessione agli endpoint SSM dai nodi ibridi. Per un elenco degli endpoint SSM, consulta AWS Systems Manager endpoints and quotas. Sostituisci `us-west-2` nel comando seguente la AWS regione per l'attivazione ibrida AWS SSM.

```
ping ssm.us-west-2.amazonaws.com
```

Visualizza lo stato della connessione delle istanze SSM registrate

Puoi controllare lo stato della connessione delle istanze registrate con le attivazioni ibride SSM con il seguente comando CLI. AWS Sostituisci l'ID della macchina con quello della macchina dell'istanza.

```
aws ssm get-connection-status --target mi-012345678abcdefgh
```

Mancata corrispondenza del checksum CLI di installazione SSM

Durante l'esecuzione di `nodeadm install`, se riscontri un problema relativo alla mancata corrispondenza del checksum `ssm-setup-cli`, dovresti confermare che non ci siano installazioni SSM precedenti esistenti sul tuo host. Se sul tuo host sono presenti installazioni SSM precedenti, rimuovile ed esegui di nuovo `nodeadm install` per risolvere il problema.

```
Failed to perform agent-installation/on-prem registration: error while verifying
 installed ssm-setup-cli checksum: checksum mismatch with latest ssm-setup-cli.
```

**`InvalidActivation`** SSM

Se riscontri un errore durante la registrazione della tua istanza con AWS SSM, conferma che i campi e e sono `region` correttiactivationCode. `activationId` `nodeConfig.yaml` La AWS regione del cluster EKS deve corrispondere alla regione dell'attivazione ibrida SSM. Se questi valori non sono configurati correttamente, potresti visualizzare un errore simile al seguente.

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 InvalidActivation
```

**`ExpiredTokenException`** SSM: Il token di sicurezza incluso nella richiesta è scaduto

Se l'agente SSM non è in grado di aggiornare le credenziali, potresti visualizzare un `ExpiredTokenException`. In questo scenario, se riesci a connetterti agli endpoint SSM dai tuoi nodi ibridi, potresti dover riavviare l'agente SSM per forzare l'aggiornamento delle credenziali.

```
"msg":"Command failed","error":"operation error SSM: DescribeInstanceInformation, https
 response error StatusCode: 400, RequestID: eee03a9e-f7cc-470a-9647-73d47e4cf0be, api
 error ExpiredTokenException: The security token included in the request is expired"
```

Errore SSM nell'esecuzione del comando registra computer

Se vedi un errore durante la registrazione del computer con SSM, potresti dover eseguire di nuovo `nodeadm install` per assicurarti che tutte le dipendenze SSM siano installate correttamente.

```
"error":"running register machine command: , error: fork/exec /opt/aws/ssm-setup-cli:
 no such file or directory"
```

**`ActivationExpired`** SSM

Durante l'esecuzione di `nodeadm init`, se vedi un errore durante la registrazione dell'istanza con SSM a causa di un'attivazione scaduta, devi creare una nuova attivazione ibrida SSM, aggiornare `nodeConfig.yaml` con `activationCode` e `activationId` della nuova attivazione ibrida SSM ed eseguire di nuovo `nodeadm init`.

```
"msg":"Command failed","error":"SSM activation expired. Please use a valid activation"
```

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 ActivationExpired
```

SSM non è riuscito ad aggiornare le credenziali memorizzate nella cache

Se riscontri un errore nell'aggiornamento delle credenziali memorizzate nella cache, il file `/root/.aws/credentials` potrebbe essere stato eliminato sul tuo host. Controlla innanzitutto l'attivazione ibrida SSM e assicurati che sia attiva e che i nodi ibridi siano configurati correttamente per utilizzare l'attivazione. Controlla l'accesso dell'agente SSM a `/var/log/amazon/ssm` ed esegui di nuovo il comando `nodeadm init` una volta risolto il problema sul lato SSM.

```
"Command failed","error":"operation error SSM: DescribeInstanceInformation, get
 identity: get credentials: failed to refresh cached credentials"
```

Pulisci la SSM

Per rimuovere l'agente SSM dal tuo host, puoi eseguire i seguenti comandi.

```
dnf remove -y amazon-ssm-agent
sudo apt remove --purge amazon-ssm-agent
snap remove amazon-ssm-agent
rm -rf /var/lib/amazon/ssm/Vault/Store/RegistrationKey
```

Risoluzione problemi AWS IAM Roles Anywhere

Se utilizzi AWS IAM Roles Anywhere per le credenziali dei tuoi nodi ibridi, tieni presente le seguenti directory e artefatti che vengono installati sui tuoi nodi ibridi da. `nodeadm` Per ulteriori informazioni

sulla risoluzione dei problemi di IAM Roles Anywhere, consulta Troubleshooting IAM Roles Anywhere identity and access nella AWS IAM Roles Anywhere User Guide AWS .

| Description | Location (Ubicazione) |
|---|---|
| CLI IAM Roles Anywhere | `/usr/local/bin/aws_signing_helper` |
| Ubicazione e nome predefiniti del certificato | `/etc/iam/pki/server.pem` |
| Posizione e nome predefiniti della chiave | `/etc/iam/pki/server.key` |

IAM Roles Anywhere non è riuscito ad aggiornare le credenziali memorizzate nella cache

Se riscontri un errore nell'aggiornamento delle credenziali memorizzate nella cache, esamina il contenuto di `/etc/aws/hybrid/config` e conferma che IAM Roles Anywhere sia stato configurato correttamente nella tua configurazione `nodeadm`. Confermare l'esistenza di `/etc/iam/pki`. Ogni nodo deve avere un certificato e una chiave univoci. Per impostazione predefinita, quando si utilizza IAM Roles Anywhere come fornitore di credenziali, `nodeadm` utilizza `/etc/iam/pki/server.pem` per la posizione e il nome del certificato e `/etc/iam/pki/server.key` per la chiave privata. Potresti dover creare le directory prima di inserire i certificati e le chiavi nelle directory con `sudo mkdir -p /etc/iam/pki`. Puoi verificare il contenuto del certificato con il comando seguente.

```
openssl x509 -text -noout -in server.pem
```

```
open /etc/iam/pki/server.pem: no such file or directory
could not parse PEM data
Command failed {"error": "... get identity: get credentials: failed to refresh cached
 credentials, process provider error: error in credential_process: exit status 1"}
```

IAM Roles Anywhere non è autorizzato a eseguire **`sts:AssumeRole`**

Nei log `kubelet`, se riscontri un problema di accesso negato all'operazione `sts:AssumeRole` quando utilizzi IAM Roles Anywhere, controlla la policy di fiducia del tuo ruolo IAM dei nodi ibridi per confermare che il principale del servizio IAM Roles Anywhere sia autorizzato ad assumere il ruolo IAM dei nodi ibridi. Verifica inoltre che l'ancora di fiducia ARN sia configurato correttamente nella policy di fiducia del ruolo IAM dei nodi ibridi e che il ruolo IAM dei nodi ibridi sia aggiunto al tuo profilo IAM Roles Anywhere.

```
could not get token: AccessDenied: User: ... is not authorized to perform:
 sts:AssumeRole on resource: ...
```

IAM Roles Anywhere non è autorizzato a impostare **`roleSessionName`**

Nei log `kubelet`, se riscontri un problema di accesso negato durante l'impostazione di `roleSessionName`, conferma che l'impostazione `acceptRoleSessionName` sia su true per il tuo profilo IAM Roles Anywhere.

```
AccessDeniedException: Not authorized to set roleSessionName
```

## Risoluzione dei problemi del sistema operativo

RHEL

Errori di registrazione del gestore dei diritti o degli entitlement

Se `nodeadm install` è in esecuzione e non è possibile installare le dipendenze dei nodi ibridi a causa di problemi di registrazione degli entitlement, assicurati di aver impostato correttamente il nome utente e la password di Red Hat sull'host.

```
This system is not registered with an entitlement server
```

Ubuntu

GLIBC non trovato

Se utilizzi Ubuntu per il sistema operativo e IAM Roles Anywhere per il provider di credenziali con nodi ibridi e riscontri un problema relativo a GLIBC non trovato, puoi installare la dipendenza manualmente per risolverlo.

```
GLIBC_2.32 not found (required by /usr/local/bin/aws_signing_helper)
```

Esegui i comandi riportati qui di seguito per installare la dipendenza:

```
ldd --version
sudo apt update && apt install libc6
sudo apt install glibc-source
```

Bottlerocket

Se hai abilitato il container di amministrazione Bottlerocket, puoi accedervi con SSH per il debug avanzato e la risoluzione dei problemi con privilegi elevati. Le seguenti sezioni contengono comandi che devono essere eseguiti nel contesto dell'host Bottlerocket. Una volta che sei nel container di amministrazione, puoi eseguire `sheltie` per ottenere una shell root completa nell'host Bottlerocket.

```
sheltie
```

Puoi anche eseguire i comandi nelle seguenti sezioni dalla shell del container di amministrazione anteponendo a ogni comando il prefisso `sudo chroot /.bottlerocket/rootfs`.

```
sudo chroot /.bottlerocket/rootfs <command>
```

Utilizzo di logdog per la raccolta dei log

Bottlerocket fornisce l'utilità `logdog` per raccogliere in modo efficiente log e informazioni di sistema per la risoluzione dei problemi.

```
logdog
```

L'utilità `logdog` raccoglie i log da varie posizioni su un host Bottlerocket e li combina in un tarball. Per impostazione predefinita, il tarball viene creato in `/var/log/support/bottlerocket-logs.tar.gz` ed è accessibile dai contenitori host in `/.bottlerocket/support/bottlerocket-logs.tar.gz`.

Accesso ai log di sistema con journalctl

Puoi controllare lo stato dei vari servizi di sistema come `kubelet`, `containerd`, ecc. e visualizzarne i registri con i seguenti comandi. Il flag `-f` seguirà i log in tempo reale.

Per controllare lo stato del servizio `kubelet` e recuperare i log `kubelet`, puoi eseguire:

```
systemctl status kubelet
journalctl -u kubelet -f
```

Per controllare lo stato del servizio `containerd` e recuperare i log per l'istanza orchestrata `containerd`, puoi eseguire:

```
systemctl status containerd
journalctl -u containerd -f
```

Per controllare lo stato del servizio `host-containerd` e recuperare i log per l'istanza host `containerd`, puoi eseguire:

```
systemctl status host-containerd
journalctl -u host-containerd -f
```

Per recuperare i log per i container bootstrap e i container host, puoi eseguire:

```
journalctl _COMM=host-ctr -f
```

## Gateway Amazon EKS Hybrid Nodes

Il gateway Amazon EKS Hybrid Nodes automatizza il networking tra il VPC del cluster Amazon EKS e i pod Kubernetes in esecuzione su nodi ibridi EKS. Il gateway elimina la necessità di rendere instradabili le reti di pod locali dal VPC o di coordinare le modifiche all'infrastruttura di rete. Crea tunnel VXLAN tra i nodi EC2-based gateway nel VPC e i nodi Cilium-managed ibridi nell'ambiente locale e mantiene automaticamente le voci della tabella di routing VPC in modo che il traffico raggiunga l'istanza gateway corretta.

### Casi d'uso

Il gateway Hybrid Nodes consente i seguenti flussi di traffico tra il VPC e l'ambiente locale:

- Comunicazione dal piano di controllo al webhook: il server API Kubernetes può raggiungere gli endpoint webhook in esecuzione su nodi ibridi. Senza il gateway, i webhook sui nodi ibridi non sono raggiungibili dal piano di controllo a meno che i pod CIDR non siano resi instradabili nell'ambiente locale.
- Pod-to-pod traffico su cloud e in locale: i pod in esecuzione sui nodi EC2 nel VPC possono comunicare direttamente con i pod in esecuzione su nodi ibridi e viceversa.
- AWS connettività del servizio ai pod ibridi: AWS servizi come Application Load Balancers, Network Load Balancers e Amazon Managed Service for Prometheus possono raggiungere i pod in esecuzione su nodi ibridi.

## Architecture

Due gateway pod vengono eseguiti come distribuzione su nodi EC2 etichettati. L'elezione del Lease-based leader di Kubernetes determina quale pod è attivo. Entrambi i pod creano un'interfaccia VXLAN all'avvio ed eseguono un riconciliatore di nodi che controlla gli `CiliumNode` oggetti, in modo che lo standby sia sempre pronto per inoltrare il traffico entro 3-5 secondi in caso di failover. Solo le azioni specifiche del leader (aggiornamenti e `CiliumVTEPConfig` gestione della tabella di routing VPC) vengono trasferite quando la leadership cambia.

## Come funziona

Il gateway Hybrid Nodes utilizza quattro meccanismi per abilitare la connettività:

Tunneling VXLAN: il gateway crea un'interfaccia VXLAN (`hybrid_vxlan0`) con VNI 2 sulla porta UDP 8472 (impostazione predefinita di Cilium). Stabilisce un tunnel per ogni nodo ibrido programmando voci FDB, voci ARP e route sull'interfaccia VXLAN. Un controller di nodi controlla gli `CiliumNode` oggetti e aggiunge o rimuove automaticamente i tunnel quando i nodi ibridi entrano o escono dal cluster.

Gestione delle tabelle di routing VPC: quando il gateway diventa il leader, crea o sostituisce le route nelle tabelle di routing VPC specificate. Ogni percorso indirizza un pod ibrido CIDR all'ENI principale del leader, quindi il traffico VPC destinato ai pod ibridi viene inoltrato all'istanza gateway attiva.

Integrazione con Cilium VTEP: il gateway crea una risorsa `CiliumVTEPConfig` personalizzata che indica agli agenti Cilium sui nodi ibridi dove inviare il traffico. VPC-bound La configurazione contiene l'IP del nodo leader come endpoint del tunnel e l'indirizzo MAC dell'interfaccia VXLAN. Quando i pod ibridi inviano traffico agli indirizzi VPC, Cilium lo incapsula in un pacchetto VXLAN e lo invia al gateway.

Elezione dei leader: il gateway utilizza Kubernetes Leader Election con un modello di standby attivo. Lease-based Due gateway pod funzionano su nodi separati grazie all'antiaffinità dei pod. Entrambi i pod creano un'interfaccia VXLAN all'avvio ed eseguono un riconciliatore di nodi che mantiene le voci VTEP per tutti i nodi ibridi. Il leader esegue gli aggiornamenti della tabella di routing VPC e la configurazione VTEP di Cilium. Se il leader fallisce, lo standby rileva la scadenza del lease, acquisisce il lease ed esegue la sequenza di configurazione del leader. Il tempo di failover previsto è di circa 3-5 secondi.

## Modello di implementazione

Il gateway Hybrid Nodes funziona su istanze EC2 nel tuo VPC e viene distribuito utilizzando un grafico Helm. Il gateway supporta i seguenti obiettivi di implementazione:

- Modalità automatica EKS: crei un `NodePool` e `NodeClass` che fornisce automaticamente ai nodi gateway le etichette, i difetti e la configurazione di source/destination controllo corretti. Questa è la configurazione consigliata.
- Gruppi di nodi gestiti EKS: si crea un gruppo di nodi gestiti dedicato con l'etichetta, il taint e il source/destination check del gateway disabilitati, quindi impostati i `autoMode.enabled=false` valori Helm.

Per tutti gli obiettivi di implementazione, sono consigliati almeno due nodi per un'elevata disponibilità. Per ulteriori informazioni, consulta the section called “Nozioni di base”.

## Prezzi

Non sono previsti costi aggiuntivi per il gateway Amazon EKS Hybrid Nodes, ma ti verranno addebitati i costi di infrastruttura per la gestione del gateway, incluse le istanze EC2 e le tariffe di gestione EKS Auto Mode, se applicabili. Per ulteriori informazioni, consultare Prezzi di Amazon EKS.

## Disponibilità nelle regioni

Il gateway Amazon EKS Hybrid Nodes è disponibile in tutte le AWS regioni in cui è disponibile EKS Hybrid Nodes, ad eccezione delle regioni della Cina. Per l'elenco aggiornato delle regioni supportate, consultathe section called “Nodi ibridi”.

## Open source

La codebase del gateway Amazon EKS Hybrid Nodes è open source. Puoi visualizzare il codice sorgente, segnalare problemi e contribuire al GitHub repository.

## Considerazioni e limitazioni

Prima di distribuire il gateway Hybrid Nodes, considera quanto segue:

- Nessuna crittografia del traffico: i tunnel VXLAN creati dal gateway non crittografano il traffico. Se è necessaria la crittografia in transito tra il VPC e l'ambiente locale, utilizza un trasporto crittografato come AWS Direct Connect with MacSec o una connessione VPN. Per ulteriori informazioni, consulta Protezione dei dati in Amazon EKS.

- Cluster singolo: ogni implementazione gateway serve un singolo cluster EKS. Se disponi di più cluster con nodi ibridi, implementa un gateway separato per ogni cluster.
- È richiesto Cilium VTEP: il gateway richiede la versione EKS di Cilium CNI con supporto VTEP abilitato sui nodi ibridi. Altri plugin CNI non sono supportati.

## Fasi successive

- the section called "Nozioni di base"
- the section called "Configurazione"
- the section called "Operazioni"
- the section called "Risoluzione dei problemi"

## Inizia a usare il gateway EKS Hybrid Nodes

Questa pagina illustra i prerequisiti, la preparazione dell'ambiente, l'installazione, la verifica e la rimozione del gateway Amazon EKS Hybrid Nodes. Per un'introduzione al gateway e alla sua architettura, consultathe section called "Gateway di nodi ibridi".

Prerequisiti

Prima di installare il gateway Hybrid Nodes, verificate che l'ambiente soddisfi i seguenti requisiti:

- Cluster EKS con supporto Cilium CNI e VTEP: il cluster EKS deve utilizzare la versione EKS di Cilium come CNI sui nodi ibridi e Cilium VTEP deve essere abilitato. Per ulteriori informazioni, consulta the section called "Configura CNI per il gateway".
- AWS VPC CNI su nodi cloud: i nodi gateway e gli altri nodi cloud nel cluster devono utilizzare il VPC CNI. AWS Il gateway si basa sul VPC-native routing per inoltrare il traffico tra il VPC e il tunnel VXLAN.
- Connettività ibrida: è necessaria una connessione privata tra il VPC e l'ambiente locale. Puoi utilizzare AWS Direct Connect, AWS Site-to-Site VPN o una soluzione VPN personalizzata. Per ulteriori informazioni, consulta the section called "Preparazione della rete".
- Traffico VXLAN consentito: i gruppi di sicurezza collegati alle istanze EC2 del gateway devono consentire il traffico UDP in entrata e in uscita sulla porta 8472. Sul lato del nodo ibrido, le regole del firewall locale devono inoltre consentire il traffico della porta UDP 8472 da e verso gli indirizzi IP del nodo gateway.

- Autorizzazioni IAM per la gestione delle tabelle di routing: il gateway richiede le seguenti azioni EC2 per gestire le tabelle di route VPC:
  - `ec2:DescribeRouteTables`
  - `ec2:CreateRoute`
  - `ec2:ReplaceRoute`
  - `ec2:DescribeInstances`

  Puoi concedere queste autorizzazioni utilizzando uno dei seguenti approcci:

  EKS Pod Identity (recommended)

  Usa EKS Pod Identity per concedere le autorizzazioni solo all'account di servizio del gateway pod.

  a. Installa l'add-on EKS Pod Identity Agent sul tuo cluster se non è già installato:

  ```
  aws eks create-addon \
    --cluster-name CLUSTER_NAME \
    --addon-name eks-pod-identity-agent
  ```

  b. Crea un ruolo IAM con le autorizzazioni EC2 richieste e una politica di fiducia per EKS Pod Identity:

  ```
  cat > gateway-trust-policy.json << 'EOF'
  {
    "Version": "2012-10-17"        ,
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": { "Service": "pods.eks.amazonaws.com" },
        "Action": ["sts:AssumeRole", "sts:TagSession"]
      }
    ]
  }
  EOF

  aws iam create-role \
    --role-name EKSHybridNodesGatewayRole \
    --assume-role-policy-document file://gateway-trust-policy.json

  aws iam put-role-policy \
    --role-name EKSHybridNodesGatewayRole \
  ```

```
  --policy-name HybridNodesGatewayRouteTable \
  --policy-document '{
    "Version": "2012-10-17"        ,
    "Statement": [{
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeRouteTables",
        "ec2:CreateRoute",
        "ec2:ReplaceRoute",
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    }]
  }'
```

c. Crea un'associazione di identità pod che colleghi il ruolo IAM all'account del servizio gateway:

```
aws eks create-pod-identity-association \
  --cluster-name CLUSTER_NAME \
  --namespace eks-hybrid-nodes-gateway \
  --service-account eks-hybrid-nodes-gateway \
  --role-arn arn:aws:iam::ACCOUNT_ID:role/EKSHybridNodesGatewayRole
```

Node IAM role

Allega una policy in linea o gestita con le autorizzazioni richieste al ruolo IAM associato al profilo di istanza dei nodi del gateway:

```
aws iam put-role-policy \
  --role-name NODE_ROLE_NAME \
  --policy-name HybridNodesGatewayRouteTable \
  --policy-document '{
    "Version": "2012-10-17"        ,
    "Statement": [{
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeRouteTables",
        "ec2:CreateRoute",
        "ec2:ReplaceRoute",
        "ec2:DescribeInstances"
      ],
```

```
      "Resource": "*"
    }]
}'
```

Tutti i pod sui nodi del gateway avranno queste autorizzazioni.

- EKS Auto Mode (se si utilizza la modalità automatica per i nodi gateway): se si prevede di utilizzare EKS Auto Mode per il provisioning dei nodi gateway, la modalità automatica deve essere abilitata sul cluster EKS. Per ulteriori informazioni, consulta Abilitare la modalità EKS Auto.

### Preparare i nodi del gateway

Il gateway richiede almeno due nodi EC2 per l'alta disponibilità. Sono supportate due opzioni di configurazione dei nodi per il gateway:

- the section called "Modalità automatica EKS (consigliata)"— Il provisioning dei nodi viene eseguito automaticamente utilizzando un comando `NodePool` e`NodeClass`. Source/destination check, labels e taint sono tutti configurati in modo dichiarativo.
- the section called "Gruppi di nodi gestiti"— Il provisioning dei nodi viene effettuato utilizzando un gruppo di nodi gestiti o nodi autogestiti. Le etichette possono essere configurate tramite l'API del gruppo di nodi gestiti e il source/destination controllo può essere disabilitato utilizzando un modello di avvio personalizzato con dati utente.

#### Modalità automatica EKS (consigliata)

Quando si utilizza la modalità EKS Auto, è necessario creare una `NodePool` e `NodeClass` prima di installare il diagramma Helm. Fornisce NodePool alle istanze EC2 le etichette, le macchie e la configurazione di controllo corrette. source/destination Non è necessario effettuare il provisioning o configurare manualmente i nodi.

Applica le seguenti risorse al tuo cluster, sostituendo i valori segnaposto:

- *`YOUR_NODE_ROLE`*— Il nome del ruolo IAM del nodo utilizzato da EKS Auto Mode per il provisioning dei nodi. Questo è il ruolo che hai configurato (o creato da EKS) quando hai abilitato la modalità automatica sul cluster. Per ulteriori informazioni, consulta Creazione di un ruolo IAM per EKS Auto Mode.
- *`YOUR_CLUSTER_NAME`*— Il nome del tuo cluster EKS.

- *SUBNET_ID_1*, *SUBNET_ID_2* — ID di sottorete in diverse zone di disponibilità in cui verrà effettuato il provisioning dei nodi gateway.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: hybrid-gateway
spec:
  advancedNetworking:
    sourceDestCheck: DisabledPrimaryENI
  role: YOUR_NODE_ROLE
  securityGroupSelectorTerms:
    - tags:
        aws:eks:cluster-name: YOUR_CLUSTER_NAME
  subnetSelectorTerms:
    - id: SUBNET_ID_1
    - id: SUBNET_ID_2
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: hybrid-gateway
spec:
  template:
    metadata:
      labels:
        hybrid-gateway-node: "true"
    spec:
      expireAfter: 336h
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: hybrid-gateway
      requirements:
        - key: karpenter.sh/capacity-type
          operator: In
          values:
            - on-demand
        - key: eks.amazonaws.com/instance-category
          operator: In
          values:
            - c
```

```
            - m
            - r
        - key: eks.amazonaws.com/instance-generation
          operator: Gt
          values:
            - "4"
        - key: kubernetes.io/arch
          operator: In
          values:
            - amd64
        - key: kubernetes.io/os
          operator: In
          values:
            - linux
      taints:
        - key: hybrid-gateway-node
          effect: NoSchedule
      terminationGracePeriod: 24h0m0s
  disruption:
    budgets:
      - nodes: 10%
    consolidateAfter: 30s
    consolidationPolicy: WhenEmptyOrUnderutilized
```

Campi chiave in questa configurazione:

- `advancedNetworking.sourceDestCheck: DisabledPrimaryENI`— Disabilita il source/destination controllo EC2 sull'ENI primario del nodo in modo che il gateway possa inoltrare il traffico che non è indirizzato a se stesso.
- `taints`— La `hybrid-gateway-node: NoSchedule` contaminazione garantisce solo gateway pod con un programma di tolleranza corrispondente su questi nodi.
- `labels`— L'`hybrid-gateway-node: "true"`etichetta viene utilizzata dal selettore di nodi del grafico Helm per indirizzare i gateway pod a questi nodi.
- `nodeClassRef`— Collega il NodePool a NodeClass con la configurazione di controllo. source/destination

Gruppi di nodi gestiti

Quando si utilizzano gruppi di nodi gestiti, è necessario creare un gruppo di nodi dedicato per il gateway con le etichette e i caratteri richiesti e un modello di avvio personalizzato che disabiliti il source/destination controllo all'avvio.

Fase 1: creazione di un modello di avvio

Crea un modello di avvio con dati utente che disabiliti il source/destination controllo sull'ENI primario all'avvio dell'istanza:

```
# Create the launch template with user data to disable source/dest check
USERDATA=$(cat <<'SCRIPT' | base64 -w 0
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==BOUNDARY=="

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 60")
MAC=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/mac)
ENI_ID=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  "http://169.254.169.254/latest/meta-data/network/interfaces/macs/${MAC}/interface-
id")
REGION=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/placement/region)

aws ec2 modify-network-interface-attribute \
  --network-interface-id "$ENI_ID" \
  --no-source-dest-check \
  --region "$REGION"

--==BOUNDARY==--
SCRIPT
)

aws ec2 create-launch-template \
  --launch-template-name YOUR_CLUSTER_NAME-gateway-lt \
  --launch-template-data "{\"UserData\":\"${USERDATA}\",\"MetadataOptions\":
{\"HttpTokens\":\"required\",\"HttpPutResponseHopLimit\":2}}"
```

> **Note**
>
> Il ruolo IAM del nodo deve disporre dell'`ec2:ModifyNetworkInterfaceAttribute`autorizzazione affinché lo script di dati utente abbia esito positivo. Il formato multipart MIME garantisce che i dati utente vengano eseguiti insieme allo script bootstrap EKS.

Fase 2: Creare il gruppo di nodi gestiti

Crea un gruppo di nodi gestiti dedicato con l'etichetta del gateway, il taint e il modello di avvio del passaggio 1:

```
aws eks create-nodegroup \
  --cluster-name YOUR_CLUSTER_NAME \
  --nodegroup-name YOUR_CLUSTER_NAME-gateway-nodes \
  --subnets SUBNET_ID_1 SUBNET_ID_2 \
  --node-role YOUR_NODE_ROLE_ARN \
  --instance-types INSTANCE_TYPE \
  --ami-type AL2023_x86_64_STANDARD \
  --scaling-config desiredSize=2,maxSize=2,minSize=2 \
  --labels hybrid-gateway-node=true \
  --taints "key=hybrid-gateway-node,effect=NO_SCHEDULE" \
  --launch-template "name=YOUR_CLUSTER_NAME-gateway-lt,version=1"
```

Questo crea un gruppo di nodi gestiti a 2 nodi in cui:

- `--labels`imposta `hybrid-gateway-node=true` in modo che il selettore di nodi del grafico Helm abbia come target questi nodi.
- `--taints`aggiunge una `NoSchedule` macchia, quindi solo i gateway pod con un programma di tolleranza corrispondente su questi nodi.
- `--launch-template`allega il modello di avvio che source/destination disabilita il controllo all'avvio.

Usa le sottoreti in diverse zone di disponibilità per una disponibilità elevata.

Installa con Helm

Modalità automatica di EKS

Esegui il comando seguente per installare il gateway con EKS Auto Mode:

```
helm install eks-hybrid-nodes-gateway \
  oci://public.ecr.aws/eks/eks-hybrid-nodes-gateway \
  --version 1.0.0 \
  --namespace eks-hybrid-nodes-gateway \
  --create-namespace \
  --set vpcCIDR=VPC_CIDR \
  --set podCIDRs=POD_CIDRS \
  --set routeTableIDs=ROUTE_TABLE_IDS
```

Gruppi di nodi gestiti o nodi autogestiti

Per i gruppi di nodi gestiti o i nodi autogestiti, imposta: `autoMode.enabled=false`

```
helm install eks-hybrid-nodes-gateway \
  oci://public.ecr.aws/eks/eks-hybrid-nodes-gateway \
  --version 1.0.0 \
  --namespace eks-hybrid-nodes-gateway \
  --create-namespace \
  --set autoMode.enabled=false \
  --set vpcCIDR=VPC_CIDR \
  --set podCIDRs=POD_CIDRS \
  --set routeTableIDs=ROUTE_TABLE_IDS
```

Valori Helm richiesti

I seguenti valori sono obbligatori per tutte le installazioni:

| Valore | Description |
| --- | --- |
| `vpcCIDR` | Il blocco CIDR del tuo cluster EKS VPC (ad esempio,`10.0.0.0/16` ). Utilizzato per la configurazione Cilium VTEP in modo che i nodi ibridi VPC-bound indirizzino il traffico attraverso il gateway. |

| Valore | Description |
| --- | --- |
| `podCIDRs` | Comma-separated elenco dei POD CIDR utilizzati da Cilium sui nodi ibridi (ad esempio,). `10.100.0.0/16,10.101.0.0/16` Il gateway crea voci nella tabella di routing VPC e route VXLAN per questi CIDR. |
| `routeTableIDs` | Comma-separated elenco degli ID delle tabelle di routing VPC da programmare (ad esempio,). `rtb-0abc1234def567890,rtb-0fed9876cba543210` Il gateway crea percorsi in queste tabelle che indirizzano i CIDR dei pod ibridi all'istanza del gateway attiva. |

Per un elenco completo dei valori configurabili, vedere. the section called "Configurazione"

Verifica dell'installazione

Dopo aver installato il gateway, verificate che sia funzionante e integro.

Controlla lo stato del pod

Verifica che due gateway pod siano in esecuzione:

```
kubectl get pods -n eks-hybrid-nodes-gateway
```

Dovresti vedere un risultato simile a:

```
NAME                                        READY   STATUS    RESTARTS   AGE
eks-hybrid-nodes-gateway-5d4f6a7b8c-abc12   1/1     Running   0          2m
eks-hybrid-nodes-gateway-5d4f6a7b8c-def34   1/1     Running   0          2m
```

Controlla l'elezione del leader

Verifica che un gruppo abbia acquisito il contratto di locazione per l'elezione del leader:

```
kubectl get lease -n eks-hybrid-nodes-gateway
```

L'output mostra il leader attuale nella `HOLDER` colonna:

```
NAME                    HOLDER                                      AGE
hybrid-gateway-leader   eks-hybrid-nodes-gateway-5d4f6a7b8c-abc12   2m
```

Controlla l'endpoint di integrità

Verifica che l'endpoint di integrità stia rispondendo sul pod leader utilizzando il port-forwarding:

```
kubectl port-forward -n eks-hybrid-nodes-gateway LEADER_POD_NAME 8088:8088 &
curl -s http://localhost:8088/healthz
```

Un gateway integro restituisce una risposta HTTP 200.

Verifica le voci della tabella di routing di VPC

Nella console Amazon VPC o utilizzando la AWS CLI, verifica che le tabelle di routing VPC contengano voci per i CIDR del pod ibrido che puntano all'ENI dell'istanza del gateway leader:

```
aws ec2 describe-route-tables \
  --route-table-ids ROUTE_TABLE_ID \
  --query "RouteTables[].Routes[?DestinationCidrBlock=='POD_CIDR']"
```

Ogni pod CIDR ibrido deve avere un percorso `NetworkInterfaceId` impostato sull'ENI primario dell'istanza leader.

Disinstallazione

Per rimuovere il gateway Hybrid Nodes, esegui:

```
helm uninstall eks-hybrid-nodes-gateway --namespace eks-hybrid-nodes-gateway
```

> **Note**
>
> La disinstallazione del grafico Helm non rimuove automaticamente le voci della tabella di routing VPC create dal gateway. Dopo la disinstallazione, elimina manualmente i percorsi per i CIDR del tuo pod ibrido dalle tabelle di routing del VPC per evitare di indirizzare il traffico verso le istanze che non eseguono più il gateway. Puoi rimuovere i percorsi utilizzando la CLI: AWS
>
> ```
> aws ec2 delete-route \
>   --route-table-id ROUTE_TABLE_ID \
>   --destination-cidr-block POD_CIDR
> ```

Fasi successive

- the section called "Configurazione"— Personalizza i valori Helm, i flag CLI e i parametri di elezione dei leader.
- the section called "Operazioni"— Monitora il gateway, comprendi il comportamento del failover e pianifica la scalabilità.
- the section called "Risoluzione dei problemi"— Diagnostica e risolvi i problemi più comuni.

## Configurare CNI per il gateway Hybrid Nodes

Il gateway Amazon EKS Hybrid Nodes richiede una configurazione Cilium specifica per consentire la VXLAN-based connettività tra il tuo VPC e i nodi ibridi. È necessario abilitare il supporto Cilium VTEP (VXLAN Tunnel Endpoint) e disabilitare il proxy L7.

Prerequisiti

- Cilium è già installato sui tuoi nodi ibridi. Per le istruzioni di installazione, consulta the section called "Configurazione della CNI".

> **Note**
>
> Cilium VTEP è supportato solo sulle seguenti versioni minime di EKS Cilium:
>
> - Cilium 1.17.13-1 o versione successiva
> - Cilium 1.18.8-1 o successivo
> - Cilium 1.19.2-1 o successivo
>
> Se utilizzi una versione precedente, aggiorna Cilium prima di abilitare VTEP. Per istruzioni sull'aggiornamento, vedere. the section called "Configurazione della CNI"

Abilita VTEP e disabilita il proxy L7

Imposta la versione di Cilium ed esegui il `helm upgrade` comando per abilitare VTEP e disabilitare il proxy L7 sull'installazione esistente di Cilium. Puoi trovare l'ultima versione di patch disponibile su Amazon ECR Public: eks/cilium /cilium.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --version CILIUM_VERSION \
  --namespace kube-system \
  --reuse-values \
  --set vtep.enabled=true \
  --set l7Proxy=false
```

Al termine dell'aggiornamento, riavvia Cilium DaemonSet per applicare la nuova configurazione:

```
kubectl rollout restart daemonset/cilium -n kube-system
kubectl rollout status daemonset/cilium -n kube-system
```

Impostazioni richieste

| Impostazione | Valore | Description |
|---|---|---|
| `vtep.enabled` | `true` | Abilita il supporto VXLAN Tunnel Endpoint di Cilium. Ciò consente agli agenti Cilium sui nodi ibridi di incapsulare il VPC-bound traffico e inviarlo al gateway. |
| `l7Proxy` | `false` | Disattiva il proxy L7 di Cilium. Ciò è necessario per il corretto funzionamento di VTEP. Se il proxy L7 è abilitato, il traffico VTEP potrebbe essere intercettato e interrotto. |

Verifica la configurazione

Dopo aver aggiornato Cilium, verifica che VTEP sia abilitato e che il proxy L7 sia disabilitato:

```
kubectl get configmap cilium-config -n kube-system -o yaml | grep -E "enable-vtep|
enable-l7-proxy"
```

Dovresti vedere un output simile a:

```
enable-l7-proxy: "false"
enable-vtep: "true"
```

### Configura VPC CNI per il traffico di pod ibridi

Per impostazione predefinita, il AWS VPC CNI applica il NAT di origine (SNAT) a tutto il traffico dei pod in uscita dal nodo. Per garantire che i servizi Kubernetes ClusterIP con endpoint pod ibridi funzionino correttamente dai nodi cloud, escludi i podR ibridi da SNAT. Ciò consente al traffico DNAT di kube-proxy di utilizzare correttamente il routing VPC.

Imposta la variabile di `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` ambiente sui CIDR del tuo `aws-node` DaemonSet pod ibrido:

```
kubectl set env daemonset aws-node -n kube-system \
  AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS=POD_CIDRS
```

Sostituisci il valore con i tuoi pod ibridi CIDR effettivi (separati da virgole se multipli).

Questa impostazione aggiunge i pod ibridi CIDR alle regole ip del VPC CNI in modo che il traffico destinato ai pod ibridi venga instradato attraverso la tabella di routing principale, che contiene la route VPC verso il gateway ENI.

> **Note**
>
> Questo passaggio è necessario per il traffico del servizio ClusterIP dai cloud pod agli endpoint pod ibridi. La connettività diretta pod-to-pod tramite indirizzo IP funziona senza questa impostazione.

### Fasi successive

Dopo aver configurato Cilium e VPC CNI, procedi con l'installazione del gateway Hybrid Nodes. Per informazioni, consulta the section called “Nozioni di base”.

## Riferimento alla configurazione del gateway Amazon EKS Hybrid Nodes

Questa pagina documenta tutti i parametri configurabili per il gateway Amazon EKS Hybrid Nodes. Per le istruzioni di installazione, consulta the section called “Nozioni di base”. Per una panoramica del gateway, consultathe section called “Gateway di nodi ibridi”.

Valori del grafico Helm

La tabella seguente elenca tutti i valori del diagramma di `eks-hybrid-nodes-gateway` Helm. È possibile impostare questi valori utilizzando i `--set` flag o un `values.yaml` file personalizzato durante `helm install` o. `helm upgrade`

| Valore | Tipo | Predefinita | Campo obbligatorio | Descrizione |
|---|---|---|---|---|
| `image.repository` | stringa | `public.ecr.aws/eks/eks-hybrid-nodes-gateway` | No | Archivio di immagini del contenitore per l'immagine del gateway. L'impostazione predefinita è il registro pubblico di Amazon ECR. |
| `image.tag` | stringa | Grafico `appVersion` | No | Tag immagine. Il valore predefinito è `appVersion` definito in. `Chart.yaml` |
| `image.pullPolicy` | stringa | `IfNotPresent` | No | Politica di estrazione delle immagini. Valori validi: `Always`, `IfNotPresent` , `Never`. |
| `replicas` | numero intero | 2 | No | Numero di repliche dei pod gateway. La configurazione consigliata è costituita da due repliche per l'elevata disponibilità negli ambienti di produzione. |
| `nodeLabel` | stringa | `hybrid-gateway-node` | No | Chiave di etichetta del nodo utilizzata per selezionare i nodi gateway. I nodi devono avere questa etichetta impostata su`"true"`. |

| Valore | Tipo | Predefinita | Campo obbligatorio | Descrizione |
|---|---|---|---|---|
| `vpcCIDR` | stringa | `""` | Sì | Blocco VPC CIDR utilizzato per la configurazione VTEP di Cilium. Esempio: `10.0.0.0/16` . |
| `podCIDRs` | stringa | `""` | Sì | Comma-separated elenco dei pod CIDR utilizzati da Cilium sui nodi ibridi. Esempio: `10.85.0.0/16,10.86.0.0/16` . |
| `routeTableIDs` | stringa | `""` | Sì | Comma-separated elenco di ID delle tabelle di routing VPC da programmare con percorsi pod ibridi. Esempio: `rtb-0123456789abcdef0,rtb-0123456789abcdef1` . |
| `autoMode.enabled` | booleano | `true` | No | Abilita l'integrazione con EKS Auto Mode. Quando`true`, il grafico aggiunge un selettore di `eks.amazonaws.com/compute-type: auto` nodi e utilizza una strategia di aggiornamento continuo senza tempi di inattività. Impostato su `false` per gruppi di nodi gestiti o nodi autogestiti. |

Esempi di comandi di installazione

Modalità automatica EKS (impostazione predefinita):

```
helm install eks-hybrid-nodes-gateway oci://public.ecr.aws/eks/eks-hybrid-nodes-gateway
 \
  --version 1.0.0 \
  --namespace eks-hybrid-nodes-gateway \
  --create-namespace \
  --set vpcCIDR=VPC_CIDR \
```

```
  --set podCIDRs=POD_CIDRS \
  --set routeTableIDs=ROUTE_TABLE_IDS
```

Gruppi di nodi gestiti o nodi autogestiti:

```
helm install eks-hybrid-nodes-gateway oci://public.ecr.aws/eks/eks-hybrid-nodes-gateway
 \
  --version 1.0.0 \
  --namespace eks-hybrid-nodes-gateway \
  --create-namespace \
  --set autoMode.enabled=false \
  --set vpcCIDR=VPC_CIDR \
  --set podCIDRs=POD_CIDRS \
  --set routeTableIDs=ROUTE_TABLE_IDS
```

Configurazione automatica o configurazione del gruppo di nodi gestito

Il valore `autoMode.enabled` Helm controlla due comportamenti nel modello Deployment:

| Comportamento | **`autoMode.enabled=true`** (predefinito) | **`autoMode.enabled=false`** |
|---|---|---|
| Selettore di nodi | Aggiunge `eks.amazonaws.com/compute-type: auto` un'etichetta aggiuntiva al nodo gateway. | Utilizza solo l'etichetta del nodo gateway (`hybrid-gateway-node: "true"` ). |
| Strategia di aggiornamento continuo | `maxSurge: 1`, `maxUnavailable: 0` — assicura zero tempi di inattività durante gli aggiornamenti avviando un nuovo pod prima di terminare quello precedente. | `maxSurge: 0`, `maxUnavailable: 1` — interrompe il vecchio pod prima di avviarne uno nuovo, poiché la capacità del nodo è preimpostata. |

Quando si utilizza la modalità automatica EKS, è necessario creare un `NodePool` e `NodeClass` prima di installare il grafico Helm. Fornisce NodePool alle istanze EC2 le etichette e i colori corretti e verifica la configurazione. source/destination Per lo YAML richiesto, consulta. the section called "Nozioni di base" Quando si utilizzano gruppi di nodi gestiti o nodi autogestiti, è necessario effettuare personalmente il provisioning e l'etichettatura dei nodi prima di installare il grafico.

Per tutti gli obiettivi di distribuzione, Deployment utilizza `hostNetwork: true` e richiede la `NET_ADMIN` funzionalità. L'antiaffinità dei pod garantisce che i due gateway pod funzionino su nodi separati.

Ottimizzazione delle elezioni dei leader

Il gateway utilizza Kubernetes Lease-based Leader Election per mantenere un modello di standby attivo. I parametri predefiniti sono ottimizzati per un failover rapido:

| Parametro | Predefinita | Description |
|---|---|---|
| `--leader-election-lease-duration` | `3s` | Quanto tempo attende una persona non leader dopo l'ultimo rinnovo del contratto di locazione osservato prima di provare ad acquistare il contratto di locazione. I valori più bassi rilevano più rapidamente i guasti del leader, ma aumentano il rischio di falsi failover durante problemi transitori di rete. |
| `--leader-election-renew-deadline` | `2s` | Tempo massimo che il leader attivo impiega a cercare di rinnovare il contratto di locazione prima di rinunciare alla leadership. Deve essere inferiore alla durata del contratto di locazione. |
| `--leader-election-retry-period` | `1s` | Con quale frequenza i candidati tentano di riacquistare o rinnovare il contratto di locazione. |

Con i valori predefiniti, il tempo di failover previsto in caso di guasto del gateway attivo è di circa 3-5 secondi. Ciò include il tempo impiegato dallo standby per rilevare la scadenza del leasing, acquisire il lease ed eseguire la sequenza di configurazione leader (aggiornamenti della tabella di routing VPC e configurazione Cilium VTEP).

Quando modificare i parametri di elezione dei leader

- Aumenta la durata del leasing se osservi frequenti transizioni tra i leader causate dalla latenza di rete transitoria tra i gateway pod e il server API Kubernetes.

- Riduci la durata del leasing se hai bisogno di un rilevamento più rapido del failover e la rete tra i nodi gateway e il server API è affidabile e a bassa latenza.
- Aumenta il periodo di riprova per ridurre il carico del server API derivante dall'elezione del leader in cluster di grandi dimensioni.

> **Note**
>
> `--leader-election-renew-deadline`Deve essere sempre inferiore a. `--leader-election-lease-duration` Se la scadenza per il rinnovo supera la durata del contratto di locazione, il leader può perdere il contratto di locazione prima che possa rinnovarlo.

Argomenti correlati

- the section called “Gateway di nodi ibridi”
- the section called “Nozioni di base”
- the section called “Operazioni”
- the section called “Risoluzione dei problemi”

## Operazioni del gateway Amazon EKS Hybrid Nodes

Questa pagina descrive le operazioni del secondo giorno per il gateway Amazon EKS Hybrid Nodes, tra cui alta disponibilità, comportamento di failover, monitoraggio, scalabilità e ciclo di vita del tunnel VXLAN. Per le istruzioni di installazione, consulta the section called “Nozioni di base”.

Disponibilità e failover elevati

Il gateway Hybrid Nodes utilizza un modello di standby attivo con l'elezione del leader di Kubernetes. Lease-based Due gateway pod funzionano su nodi EC2 separati, rafforzati dall'antiaffinità dei pod. Entrambi i pod creano un'interfaccia VXLAN all'avvio ed eseguono un riconciliatore di nodi che mantiene le voci VTEP per tutti i nodi ibridi. Solo il leader pod gestisce le tabelle di routing VPC e il `CiliumVTEPConfig` CRD. Il pod di standby è sempre pronto a inoltrare il traffico entro 3-5 secondi in caso di failover perché dispone già di un set completo di ingressi al tunnel.

Sequenza di failover

Quando l'istanza del gateway attivo fallisce, si verifica la seguente sequenza:

1. Il pod di standby rileva che il lease leader è scaduto.
2. Lo standby pod acquisisce il contratto di locazione e diventa il nuovo leader.
3. Il nuovo leader esegue la sequenza di configurazione del leader:
   - Aggiorna le voci della tabella di routing VPC per indirizzare i pod CIDR ibridi verso l'ENI principale del nuovo leader.
   - Inverte la risorsa `CiliumVTEPConfig` personalizzata con l'IP del nuovo nodo leader e l'indirizzo MAC VXLAN.
4. Il traffico riprende a fluire attraverso il nuovo leader.

Poiché entrambi i pod mantengono sempre le interfacce VXLAN e le voci VTEP, il nuovo leader non deve ricreare l'interfaccia VXLAN o riprogrammare le voci del tunnel durante il failover. Sono necessari solo la tabella di routing VPC e `CiliumVTEPConfig` gli aggiornamenti.

Il tempo di failover previsto è di circa 3-5 secondi. Durante il failover, il traffico tra il VPC e i pod ibridi viene interrotto.

Indicazione sulla zona di disponibilità

Distribuisci i nodi gateway su due zone di disponibilità in modo che un guasto della zona di disponibilità non comprometta sia il leader che lo standby. Quando usi la modalità automatica EKS, configurala `NodeClass` con selettori di sottorete su più AZ. Per gruppi di nodi gestiti o nodi autogestiti, scegli nodi in diverse AZ al momento di etichettarli.

> **Note**
>
> Cross-AZ il traffico tra il gateway e altre risorse nel VPC comporta costi di trasferimento dati AWS Cross-AZ standard.

Parametri elettorali dei leader

I parametri predefiniti per l'elezione dei leader sono ottimizzati per un failover rapido:

| Parametro | Predefinita | Description |
| --- | --- | --- |
| `--leader-election-` | `3s` | Quanto tempo attende un non leader prima di tentare di acquisire il contratto di locazione dopo che il leader ha interrotto il rinnovo. |

| Parametro | Predefinita | Description |
| --- | --- | --- |
| `lease-duration` | | |
| `--leader-election-renew-deadline` | `2s` | Per quanto tempo il leader cerca di rinnovare il contratto di locazione prima di rinunciare. |
| `--leader-election-retry-period` | `1s` | Con quale frequenza i candidati tentano di riacquistare il contratto di locazione. |

La riduzione di questi valori riduce i tempi di failover ma aumenta il rischio di falsi failover nelle partizioni di rete. Per la maggior parte delle distribuzioni, le impostazioni predefinite sono appropriate. Per ulteriori informazioni, consulta the section called "Configurazione".

Gestione delle tabelle di routing in VPC

Il gateway gestisce le voci della tabella di routing VPC in modo che il traffico destinato ai pod CIDR ibridi raggiunga l'istanza del gateway attivo.

Come vengono gestiti i percorsi

Quando un gateway pod diventa il leader, crea o sostituisce le route in ogni tabella di routing VPC configurata. Ogni route imposta il CIDR di destinazione su un pod ibrido CIDR e l'obiettivo sull'ENI principale del leader. Se una rotta esiste già e punta all'ENI corretto, il gateway salta l'aggiornamento.

Durante il failover, il nuovo leader sostituisce le rotte esistenti in modo che indirizzino al proprio ENI. Questo è il meccanismo che reindirizza il traffico VPC al nuovo gateway attivo.

Esempio di immissione nella tabella delle rotte

Dopo che il gateway ha configurato le route, la tabella di routing VPC contiene voci simili alle seguenti:

| Destinazione | Target | Stato |
| --- | --- | --- |
| 10.0.0.0/16 | local | attiva |
| *HYBRID_POD_CIDR* | *eni-LEADER_ENI_ID* | attiva |

autorizzazioni IAM

Il gateway richiede le seguenti azioni IAM per gestire le tabelle di routing:

- ec2:DescribeRouteTables
- ec2:CreateRoute
- ec2:ReplaceRoute
- ec2:DescribeInstances

Associa queste autorizzazioni al ruolo IAM associato al profilo di istanza, all'identità del pod o alla configurazione IRSA dei nodi gateway.

Monitoraggio

Endpoint Health and Readiness

Il gateway espone gli endpoint di integrità e prontezza sulla porta: 8088

| Endpoint | Path | Description |
| --- | --- | --- |
| Controllo dell'integrità | /healthz | Restituisce HTTP 200 quando il processo del gateway è integro. Utilizzato dalla sonda Liveness di Kubernetes. |
| Controllo di prontezza | /readyz | Restituisce HTTP 200 quando il gateway è pronto a servire il traffico. Utilizzato dalla sonda di prontezza Kubernetes. |

Puoi interrogare questi endpoint manualmente per la diagnostica eseguendo un contenitore di debug temporaneo o tramite il port forwarding:

```
kubectl port-forward -n eks-hybrid-nodes-gateway POD_NAME 8088:8088 &
```

```
curl -s http://localhost:8088/healthz
curl -s http://localhost:8088/readyz
```

Endpoint di metriche

Il gateway espone le Prometheus-compatible metriche sulla porta `10080` lungo il percorso. `/metrics` Le seguenti metriche personalizzate sono disponibili in aggiunta alle metriche standard di controller-runtime.

Informazioni sul gateway:

| Metrica | Tipo | Description |
|---|---|---|
| `hybrid_gateway_info` | Gauge | Informazioni statiche sull'istanza del gateway. Sempre 1. Etichette:`node_ip`,`node_name` ,`vxlan_interface` ,`vpc_cidr`,`pod_cidr`. |

Nodi ibridi:

| Metrica | Tipo | Description |
|---|---|---|
| `hybrid_gateway_hybrid_nodes_configured` | Gauge | Numero attuale di nodi ibridi con voci VTEP configurate. |

Operazioni VTEP:

| Metrica | Tipo | Description |
|---|---|---|
| `hybrid_gateway_vtep_add_total` | Contatore | Operazioni di aggiunta VTEP completate con successo. |
| `hybrid_gateway_vtep_add_errors_total` | Contatore | Totale delle operazioni di aggiunta VTEP non riuscite. |
| `hybrid_gateway_vtep_remove_total` | Contatore | Totale delle operazioni di rimozione VTEP riuscite. |

| Metrica | Tipo | Description |
| --- | --- | --- |
| `hybrid_gateway_vtep_remove_errors_total` | Contatore | Totale delle operazioni di rimozione VTEP non riuscite. |

Elezione dei leader e tabelle dei percorsi:

| Metrica | Tipo | Description |
| --- | --- | --- |
| `hybrid_gateway_leader_is_active` | Gauge | 1 se questo pod è il leader attivo, 0 se lo standby. |
| `hybrid_gateway_leader_setup_duration_seconds` | Istogramma | Durata delle operazioni di configurazione del leader (tabelle di routing+ ciliumVTEPConfig) in secondi. |
| `hybrid_gateway_aws_route_table_update_total` | Contatore | Operazioni di aggiornamento della tabella di routing completate con successo. AWS |
| `hybrid_gateway_aws_route_table_update_errors_total` | Contatore | Totale delle operazioni di aggiornamento della tabella delle AWS rotte non riuscite. |
| `hybrid_gateway_aws_route_table_update_duration_seconds` | Istogramma | Durata delle operazioni di aggiornamento della tabella delle AWS rotte in secondi. |

Statistiche di rete (raccolte su richiesta per scrape):

| Metrica | Tipo | Description |
| --- | --- | --- |
| `hybrid_gateway_vxlan_rx_bytes_total` | Gauge | Byte totali ricevuti sull'interfaccia VXLAN. |

| Metrica | Tipo | Description |
| --- | --- | --- |
| `hybrid_gateway_vxlan_tx_bytes_total` | Gauge | Byte totali trasmessi sull'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_rx_packets_total` | Gauge | Pacchetti totali ricevuti sull'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_tx_packets_total` | Gauge | Pacchetti totali trasmessi sull'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_rx_dropped_total` | Gauge | Pacchetti totali persi al momento della ricezione dall'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_tx_dropped_total` | Gauge | Pacchetti totali persi durante la trasmissione dall'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_rx_errors_total` | Gauge | Errori totali di ricezione sull'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_tx_errors_total` | Gauge | Errori totali di trasmissione sull'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_interface_up` | Gauge | 1 se l'interfaccia VXLAN è attiva, 0 altrimenti. |
| `hybrid_gateway_vxlan_fdb_entries` | Gauge | Numero attuale di voci FDB sull'interfaccia VXLAN. |
| `hybrid_gateway_vxlan_route_count` | Gauge | Numero attuale di percorsi tramite l'interfaccia VXLAN. |
| `hybrid_gateway_primary_nic_rx_bytes_total` | Gauge | Byte totali ricevuti sull'interfaccia di rete principale. |
| `hybrid_gateway_primary_nic_tx_bytes_total` | Gauge | Byte totali trasmessi sull'interfaccia di rete principale. |

| Metrica | Tipo | Description |
| --- | --- | --- |
| `hybrid_gateway_primary_nic_rx_packets_total` | Gauge | Pacchetti totali ricevuti sull'interfaccia di rete principale. |
| `hybrid_gateway_primary_nic_tx_packets_total` | Gauge | Pacchetti totali trasmessi sull'interfaccia di rete principale. |
| `hybrid_gateway_primary_nic_rx_dropped_total` | Gauge | Pacchetti totali persi durante la ricezione dalla NIC principale. |
| `hybrid_gateway_primary_nic_tx_dropped_total` | Gauge | Pacchetti totali persi durante la trasmissione dalla NIC principale. |
| `hybrid_gateway_primary_nic_rx_errors_total` | Gauge | Errori totali di ricezione sulla NIC principale. |
| `hybrid_gateway_primary_nic_tx_errors_total` | Gauge | Errori totali di trasmissione sulla NIC principale. |
| `hybrid_gateway_primary_nic_info` | Gauge | Nome NIC primario. Sempre 1. Etichette :`interface_name` . |

CloudWatch Componente aggiuntivo Observability

Puoi utilizzare il componente aggiuntivo Amazon CloudWatch Observability per raccogliere i parametri e i log del gateway. Configura il componente aggiuntivo per archiviare lo spazio dei nomi del gateway () sulla porta. `eks-hybrid-nodes-gateway` `10080` Per il formato di configurazione corretto, consultate la documentazione del componente aggiuntivo collegata sopra.

### Considerazioni sul dimensionamento

Il gateway Hybrid Nodes utilizza un modello di standby attivo con elezione del leader, quindi solo un pod gestisce il traffico alla volta. La scalabilità orizzontale del gateway (aumentando il numero di repliche) può migliorare la disponibilità fornendo pod di standby aggiuntivi pronti per essere sostituiti durante il failover, ma non migliora le prestazioni o il throughput perché il traffico non è distribuito tra le repliche. Per scalare le prestazioni, esegui la scalabilità verticale scegliendo un tipo di istanza EC2 con una larghezza di banda di rete sufficiente per il volume di traffico.

### Guida al tipo di istanza

La velocità effettiva del gateway è limitata dalle prestazioni della rete delle istanze EC2. Quando selezioni un tipo di istanza, considera quanto segue:

- Larghezza di banda di rete: il gateway inoltra tutto il traffico tra il VPC e i pod ibridi. Scegli un tipo di istanza la cui larghezza di banda di rete soddisfi i requisiti di traffico di picco.
- Pacchetti al secondo (PPS): l'incapsulamento VXLAN aggiunge un sovraccarico per pacchetto. I carichi di lavoro con molti pacchetti di piccole dimensioni (ad esempio, microservizi con tassi di richiesta elevati) traggono vantaggio dai tipi di istanze con limiti PPS più elevati.
- Numero di nodi ibridi: ogni nodo ibrido aggiunge un endpoint del tunnel VXLAN attraverso il quale il gateway inoltra il traffico. Man mano che il numero di nodi ibridi aumenta, il traffico aggregato attraverso il gateway aumenta. Seleziona un tipo di istanza con una larghezza di banda di rete sufficiente per gestire i picchi di traffico interrete del cluster.

### Tipi di istanze consigliati

#### Produzione (10—100 nodi ibridi, traffico moderato)

Adatto per carichi di lavoro di produzione standard con traffico interrete costante.

| Tipo di istanza | vCPUs | Memoria | Rete | Note |
|---|---|---|---|---|
| `c6i.xlarge` | 4 | 8 GiB | Fino a 12,5 Gb/s | Buon equilibrio tra costi e prestazioni |
| `c6in.xlarge` | 4 | 8 GiB | Fino a 30 Gbps | Network-optimized; consigliato per la produzione |

| Tipo di istanza | vCPUs | Memoria | Rete | Note |
|---|---|---|---|---|
| `c7i.xlarge` | 4 | 8 GiB | Fino a 12,5 Gb/s | Ottimizzato per il calcolo di ultima generazione |
| `m6i.xlarge` | 4 | 16 GiB | Fino a 12,5 Gb/s | Adatto per la co-localizzazione di altri carichi di lavoro su nodi gateway |

High-throughput produzione (oltre 100 nodi ibridi, traffico intenso)

Per ambienti con requisiti significativi di larghezza di banda interrete, come carichi di lavoro a uso intensivo di dati o molte connessioni simultanee.

| Tipo di istanza | vCPUs | Memoria | Rete | Note |
|---|---|---|---|---|
| `c6in.2xlarge` | 8 | 16 GiB | Fino a 40 Gbps | Consigliato per produzioni ad alto rendimento |
| `c5n.2xlarge` | 8 | 21 GiB | Fino a 25 Gb/s | Previous-generation ottimizzato per la rete, conveniente |
| `c6in.4xlarge` | 16 | 32 GiB | Fino a 50 Gb/s | Velocità di trasmissione massima per carichi di lavoro molto pesanti |
| `c5n.4xlarge` | 16 | 42 GiB | Fino a 25 Gb/s | Elevato numero di vCPU per velocità di pacchetto estreme |

Monitora l'utilizzo della rete utilizzando le metriche del gateway (vedithe section called "Endpoint di metriche") e regola il tipo di istanza in base alle esigenze.

Ciclo di vita del tunnel VXLAN

Il gateway gestisce automaticamente i tunnel VXLAN verso i nodi ibridi quando entrano o escono dal cluster.

Come vengono gestiti i tunnel

Un controller di nodi controlla `CiliumNode` gli oggetti nel cluster. Il controller funziona su ogni gateway pod (non solo su quello leader) in modo che sia il leader che lo standby abbiano lo stato del tunnel aggiornato. Quando si verifica un `CiliumNode` evento, il controller verifica se il nodo è un nodo ibrido cercando l'`eks.amazonaws.com/compute-type: hybrid`etichetta.

Quando un nodo ibrido si unisce al cluster:

1. Il controller rileva il nuovo `CiliumNode` oggetto.
2. Estrae l'indirizzo IP interno del nodo e il pod CIDR dalle specifiche. `CiliumNode`
3. Programma quanto segue sull'interfaccia VXLAN:
   - Un percorso per il pod CIDR del nodo tramite l'IP del nodo tramite l'interfaccia VXLAN.
   - Una voce ARP statica che mappa l'IP del nodo su un indirizzo MAC deterministico.
   - Una voce FDB che indica al modulo VXLAN di inviare pacchetti incapsulati all'IP del nodo.

Quando un nodo ibrido lascia il cluster:

1. Il controller rileva l'`CiliumNode`eliminazione.
2. Rimuove il percorso, la voce ARP e la voce FDB per quel nodo dall'interfaccia VXLAN.

Questo ciclo di vita è completamente automatico. Non è necessario configurare manualmente i tunnel quando si aggiungono o si rimuovono nodi ibridi.

Fasi successive

- the section called “Configurazione”— Personalizza i valori Helm, i flag CLI e i parametri di elezione dei leader.
- the section called “Risoluzione dei problemi”— Diagnostica e risolvi i problemi più comuni.
- the section called “Gateway di nodi ibridi”— Torna alla pagina di panoramica.

## Risoluzione dei problemi relativi al gateway Amazon EKS Hybrid Nodes

Questa pagina fornisce una guida per la diagnosi e la risoluzione di problemi comuni con il gateway Amazon EKS Hybrid Nodes. Ogni sezione descrive un sintomo, le possibili cause, i passaggi diagnostici e le soluzioni. Per i dettagli operativi, vederethe section called “Operazioni”.

Pod su nodi ibridi non raggiungibili da VPC

I pod in esecuzione su nodi ibridi non sono raggiungibili dalle risorse del VPC, come le istanze EC2, i sistemi di bilanciamento del carico o il piano di controllo di Kubernetes.

Possibili cause:

- Le voci della tabella di routing del VPC sono mancanti o indirizzano all'ENI sbagliato.
- Il gateway leader pod non è in esecuzione o non ha completato la configurazione.
- Cilium VTEP non è abilitato o configurato sui nodi ibridi.
- Source/destination il controllo è abilitato sull'istanza EC2 del gateway.

Fasi diagnostiche:

1. Controlla le voci della tabella di routing del VPC. Verifica che i percorsi per i CIDR del tuo pod ibrido esistano e puntino all'ENI primario dell'istanza del gateway attivo:

   ```
   aws ec2 describe-route-tables \
     --route-table-ids ROUTE_TABLE_ID \
     --query "RouteTables[].Routes[?DestinationCidrBlock=='POD_CIDR']"
   ```

   Se mancano delle route, controlla i log del gateway per eventuali errori nella tabella delle route. Se le route puntano all'ENI sbagliato, è possibile che un failover non sia stato completato correttamente.

2. Controlla lo stato del gateway pod e l'elezione del leader. Verifica che due gateway pod siano in esecuzione e uno detenga il contratto di locazione leader:

   ```
   kubectl get pods -n eks-hybrid-nodes-gateway
   kubectl get lease -n eks-hybrid-nodes-gateway
   ```

   Se nessun pod detiene il contratto di locazione, vedi. the section called “Problemi relativi alle elezioni dei leader”

3. Verifica la configurazione VTEP di Cilium sui nodi ibridi. Verifica che la `CiliumVTEPConfig` risorsa esista e contenga l'IP del nodo del leader:

   ```
   kubectl get ciliumvtepconfig hybrid-gateway -o yaml
   ```

`spec.endpoints[0].tunnelEndpoint`Deve corrispondere all'indirizzo IP del nodo del gateway leader. Se la risorsa è mancante o ha un IP obsoleto, è possibile che la configurazione del gateway non sia stata completata.

4. Controlla source/destination controlla.  source/destination Verifica che il controllo sia disabilitato sulle istanze EC2 del gateway:

```
aws ec2 describe-instance-attribute \
  --instance-id GATEWAY_INSTANCE_ID \
  --attribute sourceDestCheck
```

In caso `sourceDestCheck` affermativo`true`, disattivatelo. Consulta the section called "Nozioni di base".

Le chiamate Webhook ai nodi ibridi hanno esito negativo

Il server API Kubernetes non può raggiungere gli endpoint webhook in esecuzione su nodi ibridi. L'ammissione di Webhook richiede un timeout o restituisce errori di connessione.

Possibili cause:

- Il gateway non sta indirizzando il traffico dal piano di controllo ai pod ibridi.
- La `CiliumVTEPConfig` risorsa è mancante o ha un IP dell'endpoint obsoleto.

Fasi diagnostiche:

1. Verificare che il piano di controllo possa raggiungere l'IP del nodo gateway.  Il piano di controllo invia il traffico alla tabella delle rotte VPC, che lo inoltra all'ENI del gateway. Conferma che le voci della tabella di routing VPC siano corrette utilizzando i passaggi indicati. the section called "Pod su nodi ibridi non raggiungibili da VPC"
2. Controlla la risorsa CiliumVTepConfig.  Verifica che la risorsa esista e che `tunnelEndpoint` corrisponda all'IP del nodo del leader corrente:

```
kubectl get ciliumvtepconfig hybrid-gateway -o yaml
```

Se l'endpoint del tunnel è obsoleto (rimanda a un leader precedente), il gateway potrebbe non aver completato la sequenza di configurazione del leader. Verificate la presenza di errori nei log del gateway durante l'upsert`CiliumVTEPConfig`.

Gli aggiornamenti della tabella di routing di VPC non riescono

I log del gateway mostrano errori relativi alle operazioni della tabella di routing VPC e i percorsi per i CIDR pod ibridi non vengono creati o aggiornati.

Possibili cause:

- Il ruolo IAM del gateway non dispone delle autorizzazioni EC2 richieste.
- Gli ID delle tabelle di routing nella configurazione non sono corretti o le tabelle di route non esistono.
- Il gateway non può raggiungere l'endpoint dell'API EC2.

Fasi diagnostiche:

1. Verifica le autorizzazioni IAM.  Il gateway richiede le seguenti azioni IAM:
   - `ec2:DescribeRouteTables`
   - `ec2:CreateRoute`
   - `ec2:ReplaceRoute`
   - `ec2:DescribeInstances`

   Controlla il ruolo IAM associato al profilo di istanza del nodo del gateway o alla configurazione dell'identità del pod.
2. Controlla gli ID della tabella di routing nella configurazione.  Verifica che la variabile di `ROUTE_TABLE_IDS` ambiente contenga ID validi della tabella di routing nella distribuzione del gateway:

   ```
   kubectl get deployment eks-hybrid-nodes-gateway -n eks-hybrid-nodes-gateway -o
    jsonpath='{.spec.template.spec.containers[0].env}' | jq .
   ```

   Verifica che gli ID della tabella di routing siano presenti nel tuo VPC:

   ```
   aws ec2 describe-route-tables --route-table-ids ROUTE_TABLE_ID
   ```

3. Controlla i log del gateway per individuare eventuali errori nella tabella di routing. Cerca i messaggi di errore relativi alle operazioni della tabella di routing:

```
kubectl logs -n eks-hybrid-nodes-gateway LEADER_POD | grep -i "route table"
```

I messaggi di errore più comuni includono:

- `Failed to verify route table access`— Il gateway non è in grado di descrivere la tabella delle rotte. Controlla le autorizzazioni IAM e gli ID della tabella di routing.
- `Failed to update route tables`— Il gateway non può creare o sostituire percorsi. Controlla le autorizzazioni IAM.
- `failed to access route table`— L'ID della tabella di routing potrebbe essere errato o il ruolo IAM non è valido. `ec2:DescribeRouteTables`

I gateway pod non si avviano o non sono integri

I gateway pod sono attivi o sono in `CrashLoopBackOff` `Pending` stato oppure l'endpoint di integrità restituisce un errore. `Error`

Possibili cause:

- Le variabili di ambiente richieste (`VPC_CIDR`,`POD_CIDRS`,`ROUTE_TABLE_IDS`) non sono impostate.
- L'inoltro IP non è abilitato sul nodo gateway.
- L'etichetta del nodo o i vincoli anti-affinità impediscono la pianificazione.

Fasi diagnostiche:

1. Controlla i log dei pod. Visualizza i log del pod in errore per identificare l'errore:

```
kubectl logs -n eks-hybrid-nodes-gateway LEADER_POD
```

2. Controlla le variabili di ambiente richieste. Il gateway richiede`NODE_IP`,`VPC_CIDR`, e`POD_CIDRS`. Se ne manca uno, il gateway esce immediatamente. Verifica le specifiche del pod:

```
kubectl get pod -n eks-hybrid-nodes-gateway LEADER_POD -o
 jsonpath='{.spec.containers[0].env}' | jq .
```

- `NODE_IP`viene impostato automaticamente dalle `status.hostIP` specifiche del pod. Se è vuoto, il pod potrebbe non essere ancora pianificato su un nodo.
- `VPC_CIDR`e `POD_CIDRS` provengono dai valori di Helm. Verificate che siano impostati correttamente.

3. Controlla l'inoltro IP.  Il gateway verifica che l'inoltro IP sia abilitato all'avvio ed esce in caso contrario. Cerca il messaggio di errore `IP forwarding is not enabled` nei log del pod. Abilita l'inoltro IP sul nodo:

```
# Check current setting
cat /proc/sys/net/ipv4/ip_forward

# Enable if not set
sudo sysctl -w net.ipv4.ip_forward=1
```

   Per un'impostazione persistente, configura l'inoltro IP tramite il kubelet o aggiungi a. `net.ipv4.ip_forward=1` `/etc/sysctl.d/`

4. Controlla l'etichetta del nodo e i vincoli di pianificazione.  I gateway pod richiedono nodi con l'`hybrid-gateway-node=true`etichetta. L'antiaffinità dei pod assicura che ogni pod funzioni su un nodo separato. Se i pod lo sono`Pending`, verifica la presenza di problemi di pianificazione:

```
kubectl describe pod -n eks-hybrid-nodes-gateway LEADER_POD
```

   Cerca gli eventi che indicano nodi insufficienti, etichette mancanti o conflitti di anti-affinità.

### Problemi relativi alle elezioni dei leader

I gateway pod sono funzionanti ma nessun pod acquisisce il contratto di leadership, oppure le transizioni di leadership avvengono frequentemente.

Possibili cause:

- Mancano le autorizzazioni RBAC per gli oggetti Lease.
- La connettività di rete tra i gateway pod e il server API Kubernetes non è affidabile.

- I parametri di elezione dei leader non sono configurati correttamente.

Fasi diagnostiche:

1. Controlla l'oggetto Lease. Verifica l'esistenza del contratto di locazione e ispeziona il suo attuale titolare:

   ```
   kubectl get lease -n eks-hybrid-nodes-gateway hybrid-gateway-leader -o yaml
   ```

   Il `spec.holderIdentity` campo mostra il leader attuale. `spec.renewTime`Indica quando il contratto di locazione è stato rinnovato l'ultima volta. Se `renewTime` è obsoleto, il leader potrebbe aver perso la connettività al server API.
2. Controlla le autorizzazioni RBAC. L'account del servizio gateway necessita delle autorizzazioni per ottenere, creare e aggiornare oggetti Lease nel namespace del gateway. Verifica il ruolo e: RoleBinding

   ```
   kubectl get role -n eks-hybrid-nodes-gateway
   kubectl get rolebinding -n eks-hybrid-nodes-gateway
   ```

   Il ruolo deve includere `get`, `create`, e i `update` verbi per la `leases` risorsa nel gruppo `coordination.k8s.io` API.
3. Controlla i log dei pod per eventuali errori di leasing. Cerca gli errori di elezione dei leader nei log del pod:

   ```
   kubectl logs -n eks-hybrid-nodes-gateway LEADER_POD | grep -i "leader\|lease"
   ```

   I problemi comuni includono:

   - `Failed to acquire lease`— Il pod non può creare o aggiornare l'oggetto Lease. Controlla le autorizzazioni RBAC.
   - Frequenti `Leadership ended` seguiti da `Leader setup complete` messaggi: il leader perde e riacquista il contratto di locazione. Ciò può indicare l'instabilità della rete tra il pod e il server API. Considerate l'idea di aumentare`--leader-election-lease-duration`.
4. Controlla i parametri di elezione dei leader. Verifica i valori configurati:

   ```
   kubectl get deployment eks-hybrid-nodes-gateway -n eks-hybrid-nodes-gateway -o
    jsonpath='{.spec.template.spec.containers[0].args}'
   ```

Assicurati che `--leader-election-renew-deadline` sia inferiore a`--leader-election-lease-duration`. Se il termine di rinnovo supera la durata del contratto di locazione, il conduttore perde il contratto di locazione prima che possa rinnovarlo. Per ulteriori informazioni, consulta the section called "Ottimizzazione delle elezioni dei leader".

Messaggi di errore comuni

La tabella seguente elenca i messaggi di errore che potresti visualizzare nei log dei pod del gateway e le relative risoluzioni.

| Messaggio di errore | Causa | Risoluzione |
|---|---|---|
| `IP forwarding is not enabled` | Il parametro kernel non `net.ipv4.ip_forward` è impostato 1 sul nodo gateway. | Abilita l'inoltro IP tramite la configurazione kubelet o eseguendo. `sysctl -w net.ipv4.ip_forward=1` |
| `Failed to setup VXLAN` | Il gateway non è in grado di creare l'interfaccia di rete VXLAN. Ciò si verifica in genere quando il pod non è in grado di funzionare. `NET_ADMIN` | Verifica che le specifiche di distribuzione siano incluse `NET_ADMIN` in. `securityContext.capabilities.add` Verifica che il diagramma Helm sia distribuito correttamente. |
| `Failed to verify route table access` | Il gateway non può descrivere una o più tabelle di route VPC all'avvio. | Verifica che il ruolo IAM disponga dell'`ec2:DescribeRouteTables` autorizzazione e che gli ID della tabella di routing nella configurazione siano corretti. |
| `Failed to update route tables` | Il gateway non può creare o sostituire percorsi nelle tabelle di routing VPC. | Verifica che il ruolo IAM disponga delle `ec2:ReplaceRoute` autorizzazioni `ec2:CreateRoute` e delle autorizzazioni. |
| `Failed to create route table manager` | Il gateway non può inizializzare il client AWS | Verifica che il ruolo IAM disponga dell'`ec2:DescribeInstan` |

| Messaggio di errore | Causa | Risoluzione |
| --- | --- | --- |
| | EC2 o recuperare l'ENI primario dell'istanza. | `ces` autorizzazione e che il servizio di metadati dell'istanza (IMDS) sia accessibile. |
| `NODE_IP is required` | La variabile o il `--node-ip` flag di `NODE_IP` ambiente non è impostato . | Verifica che le specifiche del pod siano `NODE_IP` impostate `status.hostIP` utilizzando un`fieldRef`. Verifica che il diagramma Helm sia distribuito correttamente. |
| `Invalid NODE_IP` | Il valore fornito non `NODE_IP` è un indirizzo IP valido. | Controlla il valore della variabile di `NODE_IP` ambiente nelle specifiche del pod. |
| `pod-cidrs and vpc-cidr are required` | La variabile di `VPC_CIDR` ambiente `POD_CIDRS` or è vuota. | Imposta i valori `podCIDRs` e `vpcCIDR` Helm durante l'installazione. |
| `No valid route table IDs provided` | Il `ROUTE_TABLE_IDS` valore è stato impostato ma non contiene ID validi della tabella di routing dopo l'analisi. | Controlla il valore `routeTableIDs` Helm per eventuali errori di formattaz ione. Gli ID della tabella di routing devono essere separati da virgole (ad esempio,). `rtb-abc123,rtb-def 456` |
| `Failed to auto-dete ct AWS region` | Il gateway non può recuperare la AWS regione dai metadati delle istanze EC2. | Verifica che il servizio di metadati dell'istanza (IMDS) sia accessibile. In alternativa, imposta il `--aws-region` flag o la variabile di `AWS_REGION` ambiente in modo esplicito. |

| Messaggio di errore | Causa | Risoluzione |
| --- | --- | --- |
| `Failed to auto-detect AWS instance ID` | Il gateway non può recuperare l'ID dell'istanza dai metadati dell'istanza EC2. | Verifica che il servizio di metadati dell'istanza (IMDS) sia accessibile. In alternativa, imposta il `--aws-instance-id` flag o la variabile di `AWS_INSTANCE_ID` ambiente in modo esplicito. |
| `CiliumNode has no internal IP` | `CiliumNode` L'oggetto di un nodo ibrido non ha un indirizzo IP interno nelle sue specifiche. | Verifica che il nodo ibrido sia registrato correttamente e che l'agente Cilium sia in esecuzione. Controlla la `CiliumNode` risorsa per il nodo. |
| `CiliumNode <name> has no pod CIDRs allocated` | `CiliumNode` L'oggetto di un nodo ibrido non ha CIDR pod allocati da Cilium IPAM. | Verifica che Cilium IPAM sia configurato correttamente sul nodo ibrido. Controlla la `CiliumNode` risorsa per lo stato IPAM del nodo. |
| `Failed to upsert CiliumVTEPConfig` | Il gateway non può creare o aggiornare la risorsa `CiliumVTEPConfig` personalizzata. | Verifica che il CRD sia installato nel cluster e che l'account del servizio del gateway disponga delle autorizzazioni per gestire `CiliumVTEPConfig` le risorse. |
| `Unable to create manager` | L'inizializzazione del controller-runtime manager non è riuscita. | Controlla i log del pod per un contesto aggiuntivo. Le cause più comuni includono kubeconfig non valido o l'impossibilità di raggiungere il server API Kubernetes. |
| `Failed to add gateway setup` | Il runnable eletto dal leader non può essere registrato con il controller manager. | Si tratta in genere di un errore interno. Controlla i log completi del pod per ulteriori informazioni e segnala il problema nel GitHub repository. |

| Messaggio di errore | Causa | Risoluzione |
| --- | --- | --- |
| `Unable to create Node controller` | Il CiliumNode riconcili atore non può essere registrato presso il controller manager. | Controlla i log del pod per un contesto aggiuntivo. Verifica che il CiliumNode CRD sia installato nel cluster. |
| `Problem running manager` | Il controller manager è uscito in modo imprevisto. | Controlla nei log del pod l'errore sottostante. Le cause più comuni includono la perdita di connettività al server API Kubernetes o un conflitto di porte nelle metriche o negli indirizzi di bind di health probe. |
| `failed to access route table <id>` | Il gateway non è in grado di descrivere una tabella di routing VPC specifica durante il controllo di verifica dell'avvio. | Verifica che il ruolo IAM disponga `ec2:DescribeRouteTables` dell'autorizzazione e che l'ID della tabella di routing sia corretto. La tabella di routing deve esistere nella stessa regione dell'istanza del gateway. |

Argomenti correlati

- the section called "Gateway di nodi ibridi"— Panoramica dell'architettura del gateway e dei casi d'uso.
- the section called "Nozioni di base"— Prerequisiti e istruzioni di installazione.
- the section called "Configurazione"— Riferimento completo per i valori Helm, i flag CLI e le variabili di ambiente.
- the section called "Operazioni"— Guida al monitoraggio, al comportamento del failover e alla scalabilità.

# Utilizzo dell'archiviazione di dati delle applicazioni per il tuo cluster

Puoi utilizzare una gamma di servizi di AWS storage con Amazon EKS per le esigenze di storage delle tue applicazioni. Grazie a un' AWS ampia gamma di driver CSI (Container Storage Interface) supportati, puoi utilizzare facilmente Amazon EBS, Amazon S3, Amazon S3 Files, Amazon EFS, Amazon FSx e Amazon File Cache per le esigenze di storage delle tue applicazioni in esecuzione su Amazon EKS. Per gestire i backup del tuo cluster Amazon EKS, consulta Supporto di AWS backup per Amazon EKS.

Questo capitolo illustra le opzioni di archiviazione per i cluster Amazon EKS.

Argomenti


- Utilizzare il volume di archiviazione Kubernetes con Amazon EBS
- Instance Store Amazon EC2 per lo storage di volumi Kubernetes
- Utilizzo dell'archiviazione di file system elastici con Amazon EFS
- Usa lo storage del file system Amazon S3 con il driver CSI Amazon EFS
- Usa l'archiviazione delle app ad alte prestazioni con Amazon FSx per Lustre
- Usa lo storage delle app ad alte prestazioni con FSx for ONTAP NetApp
- Usa l'archiviazione di dati con Amazon FSx per OpenZFS
- Riduci al minimo la latenza con Amazon File Cache
- Accedere agli oggetti Amazon S3 con il driver CSI Mountpoint per Amazon S3
- Abilitare la funzionalità snapshot per i volumi CSI


## Utilizzare il volume di archiviazione Kubernetes con Amazon EBS

> **Note**
>
> Novità: la modalità automatica di Amazon EKS automatizza le attività di routine per l'archiviazione a blocchi. Informazioni su come the section called "Implementazione di un carico di lavoro stateful".

Il driver Container Storage Interface (CSI) di Amazon Elastic Block Store (Amazon EBS) gestisce il ciclo di vita dei volumi Amazon EBS come archiviazione per i volumi Kubernetes creati. Il driver CSI di Amazon EBS crea volumi Amazon EBS per questi tipi di volumi Kubernetes: volumi generici effimeri e volumi persistenti.

## Considerazioni

- Non è necessario installare il controller CSI di Amazon EBS sui cluster della modalità automatica EKS.
- Non è possibile montare volumi Amazon EBS sui pod Fargate.
- Puoi eseguire il controller Amazon EBS CSI sui nodi Fargate, ma sul nodo Amazon EBS CSI `DaemonSet` può essere eseguito solo su istanze Amazon EC2.
- I volumi Amazon EBS e il driver CSI di Amazon EBS non sono compatibili con Amazon EKS Hybrid Nodes.
- Verrà fornito supporto per la versione più recente del componente aggiuntivo e per una versione precedente. Le correzioni per bug o vulnerabilità rilevate nell'ultima versione verranno ripristinate nella versione precedente come nuova versione secondaria.
- La modalità automatica EKS richiede classi di archiviazione per utilizzare `ebs.csi.eks.amazonaws.com` come provisioner. Il driver CSI di Amazon EBS standard (`ebs.csi.aws.com`) gestisce i propri volumi separatamente. Per utilizzare i volumi esistenti con la modalità automatica EKS, eseguirne la migrazione tramite snapshot del volume verso una classe di archiviazione che utilizza il provisioner della modalità automatica.

> **Important**
>
> Per utilizzare la funzionalità snapshot del driver CSI di Amazon EBS, è prima necessario installare il controller di snapshot CSI. Per ulteriori informazioni, consulta the section called "Controller snapshot CSI".

## Prerequisiti

- Un cluster esistente. Per visualizzare la versione della piattaforma richiesta, esegui il comando seguente.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- Il driver EBS CSI richiede le autorizzazioni IAM. AWS
  - AWS suggerisce di utilizzare EKS Pod Identities. Per ulteriori informazioni, consulta the section called "Panoramica sulla configurazione delle associazioni EKS Pod Identity".
  - Per ulteriori informazioni sui ruoli IAM per gli account di servizio, consultare the section called "Gestore di identità OIDC IAM".

## Passaggio 1: Creazione di un ruolo IAM

Il plug-in Amazon EBS CSI richiede le autorizzazioni IAM per effettuare chiamate alle AWS API per tuo conto. Se non viene soddisfatto questo prerequisito, il tentativo di installare il componente aggiuntivo e l'esecuzione di `kubectl describe pvc` riporteranno `failed to provision volume with StorageClass` insieme a un errore `could not create volume in EC2: UnauthorizedOperation`. Per ulteriori informazioni, consulta Configurare le autorizzazioni del driver su. GitHub

> **Note**
>
> I pod avranno accesso alle autorizzazioni assegnate al ruolo IAM, a meno che non si blocchi l'accesso a IMDS. Per ulteriori informazioni, consulta the section called "Best practice".

La procedura seguente mostra come creare un ruolo IAM e allegarvi la policy AWS gestita. Per implementare questa procedura, è possibile utilizzare uno dei seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

> **Note**
>
> È possibile creare una policy autogestita con ulteriori autorizzazioni limitate. Rivedi `AmazonEBSCSIDriverPolicyV2` e crea una policy IAM personalizzata con autorizzazioni

ridotte. Se stai migrando da`AmazonEBSCSIDriverPolicy`, vedi EBS CSI Driver policy migration.

> **Note**
>
> I passaggi specifici di questa procedura sono stati scritti per l'utilizzo del driver come componente aggiuntivo di Amazon EKS. Per utilizzare il driver come componente aggiuntivo autogestito occorre una procedura differente. Per ulteriori informazioni, consulta Configurare le autorizzazioni del driver su. GitHub

### `eksctl`

1. Crea un ruolo IAM e allega una policy. AWS mantiene una policy AWS gestita oppure puoi creare una tua policy personalizzata. È possibile creare un ruolo IAM e allegare la policy AWS gestita con il seguente comando. Sostituisci *`my-cluster`* con il nome del cluster. Il comando distribuisce uno AWS CloudFormation stack che crea un ruolo IAM e vi allega la policy IAM.

```
eksctl create iamserviceaccount \
        --name ebs-csi-controller-sa \
        --namespace kube-system \
        --cluster my-cluster \
        --role-name AmazonEKS_EBS_CSI_DriverRole \
        --role-only \
        --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicyV2 \
        --approve
```

2. È possibile saltare questo passaggio se non si utilizza una chiave KMS personalizzata. Nel caso se ne utilizzi una per la crittografia dei volumi Amazon EBS, personalizzare il ruolo IAM secondo necessità. Ad esempio, completa le seguenti operazioni:

   a. Copia e incolla il codice seguente in un nuovo file `kms-key-for-encryption-on-ebs.json`. Sostituisci *`custom-key-arn`* con l'ARN della chiave KMS personalizzata.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
```

```
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

b. Crea la policy. È possibile modificare *KMS_Key_For_Encryption_On_EBS_Policy* con un nome diverso. Se esegui questa operazione, tuttavia, assicurati di modificare il nome anche nelle fasi successive.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Collega la policy IAM al ruolo con il comando seguente. Sostituisci *111122223333* con l'ID del tuo account.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
```

```
    --role-name AmazonEKS_EBS_CSI_DriverRole
```

### Console di gestione AWS

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Select trusted entity (Seleziona entità attendibile), esegui le operazioni seguenti:
   a. Nella sezione Tipo di identità attendibile, scegli Identità Web.
   b. Per Identity provider (Provider di identità), scegli URL del provider OpenID Connect per il cluster (come mostrato nella scheda Overview (Panoramica) in Amazon EKS).
   c. Per Pubblico, scegli `sts.amazonaws.com`.
   d. Scegli Successivo.
5. Nella pagina Add permissions (Aggiungi autorizzazioni), esegui le operazioni seguenti:
   a. Nella casella Filtra policy, inserisci `AmazonEBSCSIDriverPolicyV2`.
   b. Seleziona la casella di controllo a sinistra della `AmazonEBSCSIDriverPolicyV2` restituita dalla ricerca.
   c. Scegli Successivo.
6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio *`AmazonEKS_EBS_CSI_DriverRole`*.
   b. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   c. Scegli Crea ruolo.
7. Dopo aver creato il ruolo, sceglilo nella console in modo da aprirlo per la modifica.
8. Scegli la scheda Relazioni di attendibilità e quindi Modifica policy di attendibilità.
9. Trova la riga simile alla seguente:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Inserisci una virgola alla fine della riga precedente, quindi aggiungine una seguente dopo quella precedente. Sostituiscila *region-code* con la AWS regione in cui si trova il cluster. Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* con l'ID del gestore OIDC del cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. Scegli Aggiorna policy per concludere.
11. Se utilizzi una chiave KMS personalizzata per la crittografia dei volumi Amazon EBS, personalizza il ruolo IAM secondo necessità. Ad esempio, completa le seguenti operazioni:
    a. Nel pannello di navigazione a sinistra, seleziona Policy.
    b. Nella pagina Policy, scegli Crea policy.
    c. Nella pagina Crea policy, scegli la scheda JSON.
    d. Copia e incolla il codice seguente nell'editor, sostituendo *custom-key-arn* con l'ARN della chiave KMS personalizzata.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
```

```
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

e. Scegli Successivo: Tag.

f. Nella pagina Add tags (optional) (Aggiungi tag, facoltativo), seleziona Next: Review (Successivo: esamina).

g. Per Nome immettere un nome univoco per la policy (ad esempio *KMS_Key_For_Encryption_On_EBS_Policy*).

h. Scegli Crea policy.

i. Nel pannello di navigazione a sinistra, seleziona Ruoli.

j. Scegli ***AmazonEKS_EBS_CSI_DriverRole*** nella console per aprirlo e modificarlo.

k. Dall'elenco a discesa Aggiungi autorizzazioni, scegli Collega policy.

l. Nella casella Filtra policy, inserisci *KMS_Key_For_Encryption_On_EBS_Policy*.

m. Seleziona la casella di controllo a sinistra della *KMS_Key_For_Encryption_On_EBS_Policy* restituita dalla ricerca.

n. Scegli Collega policy.

## AWS CLI

1. Visualizzare l'URL del gestore OIDC del cluster. Sostituire *my-cluster* con il nome del cluster. Se l'output dal comando è `None`, rivedi i Prerequisiti.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

Di seguito viene riportato un output di esempio:

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Crea il ruolo IAM concedendo l'autorizzazione per l'operazione `AssumeRoleWithWebIdentity`.

a. Copiare i contenuti seguenti in un file denominato `aws-ebs-csi-driver-trust-policy.json`. Sostituisci *111122223333* con l'ID del tuo account. Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* con i valori restituiti nella fase precedente.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
          },
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {
            "StringEquals": {
              "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
              "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:ebs-csi-
controller-sa"
            }
          }
        }
      ]
    }
```

b. Creare il ruolo. È possibile modificare *AmazonEKS_EBS_CSI_DriverRole* con un nome diverso. Se modifichi il valore, assicurati di modificarlo anche nelle fasi successive.

```
aws iam create-role \
      --role-name AmazonEKS_EBS_CSI_DriverRole \
      --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"
```

3. Allega una politica. AWS mantiene una policy AWS gestita oppure è possibile creare una propria policy personalizzata. Allega la policy AWS gestita al ruolo con il seguente comando.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicyV2 \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Se utilizzi una chiave KMS personalizzata per la crittografia dei volumi Amazon EBS, personalizza il ruolo IAM secondo necessità. Ad esempio, completa le seguenti operazioni:

   a. Copia e incolla il codice seguente in un nuovo file `kms-key-for-encryption-on-ebs.json`. Sostituisci *`custom-key-arn`* con l'ARN della chiave KMS personalizzata.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

   b. Crea la policy. È possibile modificare *`KMS_Key_For_Encryption_On_EBS_Policy`* con un nome diverso. Se esegui questa operazione, tuttavia, assicurati di modificare il nome anche nelle fasi successive.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Collega la policy IAM al ruolo con il comando seguente. Sostituisci *111122223333* con l'ID del tuo account.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

Ora che è stato creato il ruolo IAM del driver CSI di Amazon EBS, è possibile passare alla sezione successiva. Quando si implementa il componente aggiuntivo con il ruolo IAM, questo crea e viene configurato per utilizzare un account di servizio denominato `ebs-csi-controller-sa`. L'account di servizio è associato a un `clusterrole` Kubernetes a cui sono assegnate le autorizzazioni Kubernetes richieste.

## Fase 2: ottenere il driver CSI di Amazon EBS

Consigliamo di installare il driver CSI di Amazon EBS attraverso il componente aggiuntivo di Amazon EKS per migliorare la sicurezza e ridurre le attività. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

**Important**

Prima di aggiungere il driver Amazon EBS come componente aggiuntivo di Amazon EKS, verificare che nel cluster non sia installata una versione autogestita del driver. In tal caso, vediDisinstallazione di un driver Amazon EBS CSI autogestitosul GitHub.

**Note**

Per impostazione predefinita, il ruolo RBAC utilizzato dal CSI EBS dispone delle autorizzazioni per mutare i nodi per supportare la funzionalità di rimozione

delle macchie. A causa delle limitazioni di Kubernetes RBAC, ciò consente anche di mutare qualsiasi altro nodo del cluster. Il grafico Helm ha un parametro (`node.serviceAccount.disableMutation`) che disabilita la modifica delle autorizzazioni RBAC del nodo per l'account di servizio ebs-csi-node. Se abilitate, le funzionalità del driver, come la rimozione delle macchie, non funzioneranno.

In alternativa, se desideri un'installazione autogestita del driver Amazon EBS CSI, consulta Installazione su. https://github.com/kubernetes-sigs/aws-ebs-csi-driver/blob/master/docs/install.md GitHub

### Fase 3: implementazione di un'applicazione di esempio

Puoi implementare diverse app di esempio e modificarle in base alle tue esigenze. Per ulteriori informazioni, consulta gli esempi di Kubernetes su. GitHub

## Instance Store Amazon EC2 per lo storage di volumi Kubernetes

Il driver CSI di Amazon EC2 Instance Store è un plug-in CSI (Container Storage Interface) che consente a Kubernetes di utilizzare i volumi di instance store EC2. I volumi di Instance Store forniscono uno storage temporaneo a livello di blocco collegato fisicamente al computer host. Il driver gestisce il ciclo di vita di questi volumi di storage NVMe e li rende disponibili come volumi persistenti Kubernetes.

Il driver CSI di Amazon EC2 Instance Store interagisce con i dispositivi NVMe di backend tramite operazioni Linux locali. Non sono coinvolte chiamate AWS API, quindi non è richiesto alcun ruolo IAM.

### Considerazioni

**Important**

Il driver CSI di Amazon EC2 Instance Store cancellerà tutti i dati sui dischi temporanei durante l'installazione. Se un altro driver CSI o un altro sistema di gestione dei volumi (come LVM, file system raw, LUKS, ecc.) gestisce dischi temporanei, esegui il backup dei dati prima di installare il driver CSI di Amazon EC2 Instance Store per evitare la perdita di dati.

- Il driver CSI di Amazon EC2 Instance Store non funziona con EKS Auto Mode.
- Non puoi montare volumi di istanze Amazon EC2 su Fargate Pods.
- Il driver CSI di Amazon EC2 Instance Store non è disponibile per i seguenti tipi di istanza: C1, C3, C4, C5d, C5ad, C6gd, D2, D3, D3en, DL1, E3, Edge1gd, F1, G2, G3, G4ad, G4dn, H1, HSM1, I2, I3,, I3en, M1, M2, M3, M4, M5d, M5ad, M5dn, M6gd, P2, P3, P3dn, P4d, P4de, R3, R4, R5d, R5ad, R5dn, R6gd, T1, T2, X1 I3.metal, X1e, X2gd e Z1d e Z1d e Z1d e Z1d e Z1d e Z1d D.
- Il supporto è fornito per l'ultima versione del componente aggiuntivo e una versione precedente. Le correzioni di bug o vulnerabilità rilevate nell'ultima versione verranno riportate alla versione precedente come nuova versione secondaria.
- L'archiviazione delle istanze è effimera. I dati vengono persi quando il nodo termina. Implementa la replica o il backup a livello di applicazione per i dati critici.
- L'espansione del volume non è supportata.
- I/O La limitazione è abilitata per impostazione predefinita per tutti i volumi per garantire un'equa condivisione della larghezza di banda tra i carichi di lavoro. Per ulteriori informazioni, consulta the section called “I/O strozzamento”.
- Le istanze appena lanciate hanno aggiornato le risposte ai comandi di identificazione di NVMe rispetto alle istanze precedenti dello stesso tipo. Se i tuoi strumenti dipendono da valori di identificazione NVMe specifici, esamina la tua logica per verificarne la compatibilità. Per ulteriori informazioni, consulta the section called “Modifiche allo spazio dei nomi NVMe sulle istanze appena lanciate”.

## Prerequisiti

Prima di iniziare, assicurati di disporre di:

- Un cluster Amazon EKS esistente. Per visualizzare tutte le versioni disponibili del componente aggiuntivo, esegui il seguente comando AWS CLI:

```
aws eks describe-addon-versions --addon-name aws-ec2-local-instance-store-csi-driver
```

## Installazione del driver CSI di Amazon EC2 Instance Store

Il driver CSI di Amazon EC2 Instance Store è disponibile come componente aggiuntivo Amazon EKS. L'installazione del driver consente al cluster di utilizzare i volumi di Instance Store per lo storage su pod.

Per installare il driver, aggiungi il driver aggiuntivo Amazon EC2 Instance Store CSI al cluster. Per istruzioni, consulta Creazione di un componente aggiuntivo Amazon EKS. Per ulteriori informazioni sui componenti aggiuntivi EKS, consulta i componenti aggiuntivi di Amazon EKS.

## Implementazione di un'applicazione di esempio

Dopo aver installato il driver, puoi distribuire applicazioni di esempio per verificare che i volumi di Instance Store funzionino correttamente con il tuo cluster. Puoi implementare diverse app di esempio e modificarle in base alle tue esigenze.

## I/O strozzamento

Il driver CSI di Amazon EC2 Instance Store include la I/O limitazione integrata per garantire una condivisione equa della larghezza di banda tra i carichi di lavoro sullo stesso controller NVMe. Il throttling è abilitato di default per tutti i volumi, evitando che un singolo carico di lavoro consumi una larghezza di banda sproporzionata I/O a scapito di altri carichi di lavoro sullo stesso nodo.

### Configurazione della limitazione

È possibile configurare la limitazione a due livelli:

- StorageClass parametro: imposta il `throttling` parametro su `"true"` (impostazione predefinita) o `"false"` nella StorageClass definizione.
- Annotazione PVC: imposta l'`lis.csi.aws.com/throttling`annotazione `"false"` su `"true"` o singolarmente. PersistentVolumeClaims Questa impostazione ha la precedenza sull'impostazione quando presente. StorageClass

La tabella seguente mostra come interagiscono il StorageClass parametro e l'annotazione PVC:

| Annotazione PVC | StorageClass Parametro | Risultato |
|---|---|---|
| Non impostato | Non impostato | Limitazione attivata (impostazione predefinita) |
| Non impostato | `"false"` | Limitazione disattivata |
| `"false"` | `"true"`o non impostato | Throttling disabilitato |
| `"true"` | `"false"` | Throttling abilitato |

## Creazione di un sistema senza limitazioni StorageClass

Per disabilitare la limitazione per tutti i volumi in un: StorageClass

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ec2-instance-store-unthrottled
provisioner: lis.csi.aws.com
parameters:
  throttling: "false"
volumeBindingMode: WaitForFirstConsumer
```

## Disattivazione della limitazione per un singolo PVC

Per disabilitare la limitazione su un singolo PC mentre si utilizza un limitatore: StorageClass

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: high-perf-pvc
  annotations:
    lis.csi.aws.com/throttling: "false"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: ec2-instance-store-sc
  resources:
    requests:
      storage: 100Gi
```

## Considerazioni sulla limitazione

- Il throttling è abilitato di default per fornire prestazioni prevedibili per tutti i carichi di lavoro che condividono un controller NVMe.
- La disabilitazione della limitazione su un volume consente di utilizzare una maggiore I/O larghezza di banda, il che può influire sulle prestazioni di altri volumi limitati sullo stesso controller.
- Valuta la possibilità di isolare i carichi di lavoro senza limitazioni su nodi dedicati.
- Per i RAID-0 volumi che si estendono su più controller, la limitazione viene applicata a tutte le partizioni membri.

- Lo stato di limitazione persiste anche dopo i riavvii dei nodi.

## Modifiche allo spazio dei nomi NVMe sulle istanze appena lanciate

Le istanze EC2 appena lanciate con archiviazione locale hanno aggiornato le risposte ai comandi NVMe Identify Controller e Identify Namespace. Queste modifiche supportano la gestione dello spazio dei nomi NVMe sullo storage delle istanze locali EC2. Le istanze in esecuzione esistenti non vengono influenzate e i valori rimangono stabili per tutto il ciclo di vita di qualsiasi istanza.

### Valori dei campi aggiornati

I seguenti campi sono stati modificati sulle istanze appena lanciate rispetto alle istanze precedenti dello stesso tipo:

### Identifica Controller () **`nvme id-ctrl`**

- Bit 3 di OACS: cambiato da 0 a 1
- Bit 8 di OAES: cambiato da 0 a 1
- NN: cambiato da 1 a un valore <= 256
- TNVMCAP: cambiato da 0 alla dimensione effettiva della memoria locale

### Identifica lo spazio dei nomi () **`nvme id-ns`**

- NVMCAP: cambiato da 0 alla dimensione effettiva dello storage locale
- NGUID: cambiato da zero a un NGUID valido

Tutti i valori aggiornati sono conformi alla specifica NVMe.

### Ambito delle modifiche

Queste modifiche si applicano esclusivamente allo storage delle istanze locali EC2. I volumi Amazon EBS non sono interessati. Per distinguere tra volumi EBS e storage a istanze locali, esegui il seguente comando:

```
sudo nvme list
```

Nell'output:

- `Amazon Elastic Block Store=` volume EBS
- `Amazon EC2 NVMe Instance Storage=` archiviazione locale dell'istanza

### Recupero dei valori identificativi NVMe

Per interrogare questi valori utilizzando: `nvme-cli`

```
# Identify Controller
sudo nvme id-ctrl /dev/nvme1

# Identify Namespace
sudo nvme id-ns /dev/nvme1n1
```

### Operazione richiesta

Nessuna operazione richiesta. Queste modifiche sono compatibili con le versioni precedenti e seguono le specifiche NVMe. Tuttavia, se l'applicazione o lo strumento sono convalidati o dipendono da valori specifici dei comandi di identificazione NVMe, rivedi la logica per garantire la compatibilità con i valori aggiornati sulle istanze appena lanciate.

## Utilizzo dell'archiviazione di file system elastici con Amazon EFS

Amazon Elastic File System (Amazon EFS) fornisce un'archiviazione di file serverless e completamente elastica in modo da poter condividere i dati dei file senza dover fornire o gestire la capacità e le prestazioni di archiviazione. Il driver Amazon EFS Container Storage Interface (CSI) consente ai cluster Kubernetes in esecuzione di montare i file system AWS Amazon EFS come volumi persistenti. Questo argomento illustra come implementare il driver CSI per Amazon EBS nel cluster Amazon EKS.

### Considerazioni

- Il driver CSI per Amazon EFS non è compatibile con le immagini di container basate su Windows.
- Non è possibile utilizzare il provisioning dinamico dei volumi persistenti con nodi Fargate, ma è possibile utilizzare il provisioning statico.
- Il provisioning dinamico richiede un driver versione 1.2 o successiva. È possibile eseguire il provisioning statico di volumi persistenti utilizzando la versione `1.1` del driver su qualsiasi versione del cluster Amazon EKS supportata (consultare la pagina Amazon EKS supported versions).

- La versione 1.3.2 o successiva di questo driver supporta l'architettura Arm64, incluse le istanze basate su Graviton di Amazon EC2.
- La versione 1.4.2 o successiva di questo driver supporta l'uso di FIPS per il montaggio di file system.
- Osserva le quote delle risorse per Amazon EFS. Per ulteriori informazioni, consulta le quote di Amazon EFS.
- A partire dalla versione 2.0.0, questo driver è passato dall'utilizzo di `stunnel` a `efs-proxy` per le connessioni TLS. Quando viene utilizzato `efs-proxy`, aprirà un numero di thread pari a uno più il numero di core del nodo su cui è in esecuzione.
- Il driver CSI per Amazon EFS non è compatibile con Amazon EKS Hybrid Nodes.

## Prerequisiti

- Il driver CSI Amazon EFS necessita delle autorizzazioni AWS Identity and Access Management (IAM).
  - AWS suggerisce di utilizzare EKS Pod Identities. Per ulteriori informazioni, consulta the section called "Panoramica sulla configurazione delle associazioni EKS Pod Identity".
  - Per informazioni sui ruoli IAM per gli account di servizio e su come configurare un gestore OpenID Connect (OIDC) IAM per il cluster, consultare the section called "Gestore di identità OIDC IAM".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum`apt-get, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

> **Note**
>
> Un pod in esecuzione su Fargate monta automaticamente un file system Amazon EFS senza bisogno della procedura di installazione manuale del driver.

## Passaggio 1: Creazione di un ruolo IAM

Il driver CSI per Amazon EFS richiede le autorizzazioni IAM per interagire con il file system. Crea un ruolo IAM e allega ad esso la policy `arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy` gestita.

> **Note**
>
> Se desideri utilizzare lo storage di file system Amazon EFS e Amazon S3, devi collegare sia le policy `AmazonS3FilesCSIDriverPolicy` gestite che quelle gestite al tuo ruolo IAM. `AmazonEFSCSIDriverPolicy` Per ulteriori informazioni sullo storage del file system Amazon S3, consulta. the section called "File di Amazon S3"

Per implementare questa procedura, è possibile utilizzare uno dei seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

> **Note**
>
> I passaggi specifici di questa procedura sono stati scritti per l'utilizzo del driver come componente aggiuntivo di Amazon EKS. Per informazioni dettagliate sulle installazioni autogestite, consulta Set up driver permission su GitHub.

### `eksctl`

Se si utilizza Pod Identity

Eseguire i seguenti comandi per creare un ruolo IAM e un'associazione Pod Identity con `eksctl`. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create podidentityassociation \
    --service-account-name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --permission-policy-arns arn:aws: iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy
```

Se si utilizzano ruoli IAM per gli account di servizio

Eseguire i seguenti comandi per creare un ruolo IAM con `eksctl`. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

## Console di gestione AWS

Esegui quanto segue per creare un ruolo IAM con Console di gestione AWS.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Se si utilizza EKS Pod Identity:
      i. Nella sezione Tipo di entità affidabile, scegli  AWS servizio.
      ii. Nel menu a discesa Servizio o caso d'uso, scegliere EKS.
      iii. Nella sezione Caso d'uso, scegliere EKS - Pod Identity.
      iv. Scegli Next (Successivo).
   b. Se si utilizzano ruoli IAM per gli account di servizio:
      i. Nella sezione Tipo di identità attendibile, scegli Identità Web.
      ii. Per Identity provider (Provider di identità), scegli URL del provider OpenID Connect per il cluster (come mostrato nella scheda Overview (Panoramica) in Amazon EKS).
      iii. Per Pubblico, scegli `sts.amazonaws.com`.
      iv. Scegli Next (Successivo).
5. Nella pagina Add permissions (Aggiungi autorizzazioni), esegui le operazioni seguenti:
   a. Nella casella Filtra policy, inserisci `AmazonEFSCSIDriverPolicy`.
   b. Seleziona la casella di controllo a sinistra della `AmazonEFSCSIDriverPolicy` restituita dalla ricerca.
   c. Scegli Next (Successivo).
6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKS_EFS_CSI_DriverRole`.
   b. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   c. Scegli Crea ruolo.
7. Dopo aver creato il ruolo:

a. Se si utilizza EKS Pod Identity:

   i. Aprire la Console Amazon EKS.

   ii. Nel riquadro di navigazione a sinistra, selezionare Cluster, quindi selezionare il nome del cluster per cui configurare l'associazione EKS Pod Identity.

   iii. Scegliere la scheda Accesso.

   iv. In Associazioni Pod Identity, scegliere Crea.

   v. Scegliere il menu a discesa Ruolo IAM e selezionare il ruolo appena creato.

   vi. Scegliere il campo Namespace Kubernetes e inserire `kube-system`.

   vii. Scegliere il campo Account del servizio Kubernetes e inserire `efs-csi-controller-sa`.

   viii. Scegli Create (Crea).

   ix. Per ulteriori informazioni sulla creazione di associazioni Pod Identity, consultare the section called "Crea un'associazione Pod Identity (AWS Console)".

b. Se si utilizzano ruoli IAM per gli account di servizio:

   i. Scegliere il ruolo per aprirlo per la modifica.

   ii. Scegli la scheda Relazioni di attendibilità e quindi Modifica policy di attendibilità.

   iii. Trova la riga simile alla seguente:

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
 "sts.amazonaws.com"
```

   Aggiungi la riga seguente sopra la riga precedente. Sostituiscilo `<region-code>` con la AWS regione in cui si trova il cluster. Sostituire `<EXAMPLED539D4633E53DE1B71EXAMPLE>` con l'ID del gestore OIDC del cluster.

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
```

   iv. Modifica l'operatore `Condition` da `"StringEquals"` a `"StringLike"`.

   v. Scegli Aggiorna policy per concludere.

## AWS CLI

Esegui i seguenti comandi per creare un ruolo IAM con AWS CLI.

Se si utilizza Pod Identity

1. Creare il ruolo IAM che concede le operazioni `AssumeRole` e `TagSession` al servizio `pods.eks.amazonaws.com`.
   a. Copiare i seguenti contenuti in un file denominato `aws-efs-csi-driver-trust-policy-pod-identity.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

   b. Crea il ruolo. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-identity.json"
```

2. Allega la policy AWS gestita richiesta al ruolo con il comando seguente.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

3. Eseguire il seguente comando per creare l'associazione Pod Identity. Sostituisci <111122223333> con l'ID del tuo account.

```
aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn {arn-
aws}iam::<111122223333>:role/$role_name --namespace kube-system --service-account
 efs-csi-controller-sa
```

4. Per ulteriori informazioni sulla creazione di associazioni Pod Identity, consultare the section called "Crea un'associazione Pod Identity (AWS Console)".

Se si utilizzano ruoli IAM per gli account di servizio

1. Visualizzare l'URL del gestore OIDC del cluster. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer"
 --output text
```

   Di seguito viene riportato un output di esempio.

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

   Se l'output dal comando è `None`, rivedi i Prerequisiti.

2. Crea il ruolo IAM che concede l'autorizzazione per l'operazione `AssumeRoleWithWebIdentity`.
   a. Copia i seguenti contenuti in un file denominato `aws-efs-csi-driver-trust-policy.json`. Sostituisci `<111122223333>` con l'ID del tuo account. Sostituire `<EXAMPLED539D4633E53DE1B71EXAMPLE>` e `<region-code>` con i valori restituiti nella fase precedente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
```

```
      "Condition": {
        "StringLike": {
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:efs-csi-
*",
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. Creare il ruolo.

```
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy.json"
```

3. Allega la politica AWS gestita richiesta al ruolo con il comando seguente.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

## Fase 2: ottenere il driver CSI per Amazon EFS

Consigliamo di installare il driver Amazon EFS CSI attraverso il componente aggiuntivo di Amazon EKS. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS". Se non riesci a utilizzare il componente aggiuntivo Amazon EKS, ti consigliamo di segnalare un problema sul motivo per cui non puoi farlo all'archivio della roadmap GitHub di Containers.

> **Important**
>
> Prima di aggiungere il driver Amazon EFS come componente aggiuntivo di Amazon EKS, verificare che nel cluster non sia installata una versione autogestita del driver. In tal caso, consulta Disinstallazione del driver Amazon EFS CSI su. GitHub

In alternativa, se desideri un'installazione autogestita del driver Amazon EFS CSI, consulta Installazionesu GitHub.

### Fase 3: Creazione di un file system Amazon EFS

Per creare un file system Amazon EFS, consulta Creazione di un sistema di file Amazon EFS per Amazon EKS su GitHub.

### Fase 4: implementazione di un'applicazione di esempio

Puoi implementare diverse app di esempio e modificarle in base alle tue esigenze. Per ulteriori informazioni, consulta Esempi su. GitHub

## Usa lo storage del file system Amazon S3 con il driver CSI Amazon EFS

S3 Files è un file system condiviso che collega qualsiasi AWS elaborazione direttamente ai tuoi dati in Amazon S3. Fornisce un accesso rapido e diretto a tutti i dati S3 sotto forma di file con semantica completa del file system e prestazioni a bassa latenza, senza che i dati escano mai da S3. Ciò significa che le applicazioni, gli agenti e i team basati su file possono accedere e lavorare con i dati S3 come file system utilizzando gli strumenti da cui già dipendono. Il driver Amazon EFS Container Storage Interface (CSI) consente ai cluster Kubernetes in esecuzione di montare i file system AWS Amazon S3 come volumi persistenti a partire dalla versione 3.0.0. Questo argomento mostra come utilizzare il driver CSI di Amazon EFS per gestire i file system Amazon S3 sul cluster Amazon EKS.

### Considerazioni

- Il driver CSI Amazon EFS non è compatibile con le immagini dei Windows-based container.
- EKS Fargate non supporta i file S3.
- Il driver CSI per Amazon EFS non è compatibile con Amazon EKS Hybrid Nodes.
- Il supporto di Amazon S3 Files nel driver CSI di Amazon EFS inizia dalla versione 3.0.0.

### Prerequisiti

- Il driver CSI Amazon EFS necessita delle autorizzazioni AWS Identity and Access Management (IAM).

  - AWS suggerisce di utilizzare EKS Pod Identities. Per ulteriori informazioni, consulta the section called "Panoramica sulla configurazione delle associazioni EKS Pod Identity".
  - Per informazioni sui ruoli IAM per gli account di servizio e su come configurare un gestore OpenID Connect (OIDC) IAM per il cluster, consultare the section called "Gestore di identità OIDC IAM".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

## Fase 1: Creare ruoli IAM

Il driver CSI per Amazon EFS richiede le autorizzazioni IAM per interagire con il file system. Il driver EFS CSI utilizza due account di servizio con ruoli IAM separati:

- `efs-csi-controller-sa`— utilizzato dal controller, richiede `AmazonS3FilesCSIDriverPolicy` e`AmazonS3FilesClientFullAccess`.
- `efs-csi-node-sa`— utilizzato dal nodo daemonset, richiede:
  - `AmazonS3FilesClientFullAccess`— concede le autorizzazioni per montare e scrivere su file system S3 (,,). `s3files:ClientMount` `s3files:ClientWrite` `s3files:ClientRootAccess`
  - `AmazonS3ReadOnlyAccess`— consente lo streaming di letture direttamente dal bucket S3 per una maggiore velocità di trasmissione.

- `AmazonElasticFileSystemsUtils`— consente la pubblicazione dei log di efs-utils su CloudWatch Amazon per la visibilità delle operazioni di montaggio e una risoluzione più semplice dei problemi.

> **Note**
>
> Se desideri utilizzare sia il file system Amazon S3 che lo storage Amazon EFS, devi collegare sia le policy gestite che le policy `AmazonEFSCSIDriverPolicy` gestite al ruolo di controller. `AmazonS3FilesCSIDriverPolicy` Per ulteriori informazioni sullo storage Amazon EFS, consultathe section called "Amazon EFS".

Per implementare questa procedura, è possibile utilizzare uno dei seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

> **Note**
>
> I passaggi specifici di questa procedura sono stati scritti per l'utilizzo del driver come componente aggiuntivo di Amazon EKS. Per informazioni dettagliate sulle installazioni autogestite, consulta Set up driver permission su GitHub.

## `eksctl`

Se si utilizza Pod Identity

Esegui i seguenti comandi per creare ruoli IAM e associazioni di identità Pod con`eksctl`. Sostituisci *`my-cluster`* con il tuo valore.

```
export cluster_name=my-cluster

# Create the controller role
eksctl create podidentityassociation \
    --service-account-name efs-csi-controller-sa \
    --namespace kube-system \
```

```
    --cluster $cluster_name \
    --role-name AmazonEKS_EFS_CSI_ControllerRole \
    --permission-policy-arns arn:aws: iam::aws:policy/service-role/
AmazonS3FilesCSIDriverPolicy,arn:aws: iam::aws:policy/AmazonS3FilesClientFullAccess

# Create the node role
eksctl create podidentityassociation \
    --service-account-name efs-csi-node-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name AmazonEKS_EFS_CSI_NodeRole \
    --permission-policy-arns arn:aws: iam::aws:policy/
AmazonS3FilesClientFullAccess,arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess,arn:aws:
 iam::aws:policy/AmazonElasticFileSystemsUtils
```

Se si utilizzano ruoli IAM per gli account di servizio

Esegui i seguenti comandi per creare ruoli IAM con`eksctl`. Sostituiscilo *`my-cluster`* con il nome del cluster e *`region-code`* con il codice AWS regionale.

```
export cluster_name=my-cluster
export region_code=region-code

# Create the controller role
export controller_role_name=AmazonEKS_EFS_CSI_ControllerRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $controller_role_name \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AmazonS3FilesCSIDriverPolicy \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonS3FilesClientFullAccess \
    --approve \
    --region $region_code

# Create the node role
export node_role_name=AmazonEKS_EFS_CSI_NodeRole
eksctl create iamserviceaccount \
    --name efs-csi-node-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $node_role_name \
```

```
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonS3FilesClientFullAccess \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonElasticFileSystemsUtils \
    --approve \
    --region $region_code
```

## Console di gestione AWS

Esegui quanto segue per creare un ruolo IAM con Console di gestione AWS.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Se si utilizza EKS Pod Identity:
      i. Nella sezione Tipo di entità affidabile, scegli  AWS servizio.
      ii. Nel menu a discesa Servizio o caso d'uso, scegliere EKS.
      iii. Nella sezione Caso d'uso, scegliere EKS - Pod Identity.
      iv. Scegli Next (Successivo).
   b. Se si utilizzano ruoli IAM per gli account di servizio:
      i. Nella sezione Tipo di identità attendibile, scegli Identità Web.
      ii. Per Identity provider (Provider di identità), scegli URL del provider OpenID Connect per il cluster (come mostrato nella scheda Overview (Panoramica) in Amazon EKS).
      iii. Per Pubblico, scegli `sts.amazonaws.com`.
      iv. Scegli Next (Successivo).
5. Nella pagina Add permissions (Aggiungi autorizzazioni), esegui le operazioni seguenti:
   a. Nella casella Filtra policy, inserisci `AmazonS3FilesCSIDriverPolicy`. Seleziona la casella di controllo a sinistra della politica restituita nella ricerca. Quindi cerca e selezionaAmazonS3FilesClientFullAccess.
   b. Scegli Next (Successivo).
6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKS_EFS_CSI_ControllerRole`.

b. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.

c. Scegli Crea ruolo.

7. Dopo aver creato il ruolo:

a. Se si utilizza EKS Pod Identity:

i. Aprire la Console Amazon EKS.

ii. Nel riquadro di navigazione a sinistra, selezionare Cluster, quindi selezionare il nome del cluster per cui configurare l'associazione EKS Pod Identity.

iii. Scegliere la scheda Accesso.

iv. In Associazioni Pod Identity, scegliere Crea.

v. Scegliere il menu a discesa Ruolo IAM e selezionare il ruolo appena creato.

vi. Scegliere il campo Namespace Kubernetes e inserire `kube-system`.

vii. Scegliere il campo Account del servizio Kubernetes e inserire `efs-csi-controller-sa`.

viii. Scegli Create (Crea).

ix. Per ulteriori informazioni sulla creazione di associazioni Pod Identity, consultare the section called "Crea un'associazione Pod Identity (AWS Console)".

x. Ripeti i passaggi precedenti per creare un secondo ruolo per l'account del servizio del nodo. Nella pagina Aggiungi autorizzazioni`AmazonS3FilesClientFullAccess`, allega `AmazonS3ReadOnlyAccess` e `AmazonElasticFileSystemsUtils` invece. Quindi crea un'associazione Pod Identity con `efs-csi-node-sa` per il campo dell'account del servizio Kubernetes.

b. Se si utilizzano ruoli IAM per gli account di servizio:

i. Scegliere il ruolo per aprirlo per la modifica.

ii. Scegli la scheda Relazioni di attendibilità e quindi Modifica policy di attendibilità.

iii. Trova la riga simile alla seguente:

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
 "sts.amazonaws.com"
```

Aggiungi la riga seguente sopra la riga precedente. `<region-code>`Sostituiscilo con la AWS regione in cui si trova il cluster. Sostituire `<EXAMPLED539D4633E53DE1B71EXAMPLE>` con l'ID del gestore OIDC del cluster.

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-controller-sa",
```

iv. Scegli Aggiorna policy per concludere.

v. Ripeti i passaggi precedenti per creare un secondo ruolo per l'account del servizio del nodo. Nella pagina Aggiungi autorizzazioni`AmazonS3FilesClientFullAccess`, allega `AmazonS3ReadOnlyAccess` e `AmazonElasticFileSystemsUtils` invece. Nella politica di attendibilità, utilizzare `efs-csi-node-sa` per il valore della `:sub` condizione.

## AWS CLI

Esegui i seguenti comandi per creare ruoli IAM con AWS CLI.

Se si utilizza Pod Identity

1. Creare il ruolo IAM che concede le operazioni `AssumeRole` e `TagSession` al servizio `pods.eks.amazonaws.com`.

   a. Copiare i seguenti contenuti in un file denominato `aws-efs-csi-driver-trust-policy-pod-identity.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

   b. Crea il ruolo. Sostituisci *`my-cluster`* con il nome del cluster.

```
export cluster_name=my-cluster
export controller_role_name=AmazonEKS_EFS_CSI_ControllerRole
aws iam create-role \
  --role-name $controller_role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-
identity.json"
```

2. Allega le politiche AWS gestite richieste al ruolo del controller.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonS3FilesCSIDriverPolicy \
  --role-name $controller_role_name

aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3FilesClientFullAccess \
  --role-name $controller_role_name
```

3. Crea il ruolo IAM del nodo utilizzando la stessa policy di fiducia.

```
export node_role_name=AmazonEKS_EFS_CSI_NodeRole
aws iam create-role \
  --role-name $node_role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-
identity.json"
```

4. Allega le politiche AWS gestite richieste al ruolo del nodo.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3FilesClientFullAccess \
  --role-name $node_role_name

aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess \
  --role-name $node_role_name

aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonElasticFileSystemsUtils \
  --role-name $node_role_name
```

5. Esegui i seguenti comandi per creare le associazioni Pod Identity. Sostituisci <111122223333> con l'ID del tuo account.

```
aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn {arn-
aws}iam::<111122223333>:role/$controller_role_name --namespace kube-system --service-
account efs-csi-controller-sa
aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn
 {arn-aws}iam::<111122223333>:role/$node_role_name --namespace kube-system --service-
account efs-csi-node-sa
```

6. Per ulteriori informazioni sulla creazione di associazioni Pod Identity, consultare the section called "Crea un'associazione Pod Identity (AWS Console)".

Se si utilizzano ruoli IAM per gli account di servizio

1. Visualizzare l'URL del gestore OIDC del cluster. Sostituisci *my-cluster* con il nome del cluster.

```
export cluster_name=my-cluster
aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer"
 --output text
```

   Di seguito viene riportato un output di esempio.

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

   Se l'output dal comando è None, rivedi i Prerequisiti.

2. Crea il ruolo IAM per l'account del servizio controller.

   a. Copia i seguenti contenuti in un file denominato `controller-trust-policy.json`. Sostituisci <111122223333> con l'ID del tuo account. Sostituire <EXAMPLED539D4633E53DE1B71EXAMPLE> e <region-code> con i valori restituiti nella fase precedente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws: iam::<111122223333>:oidc-provider/
oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>"
            },
```

```
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud": "sts.amazonaws.com",
                    "oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-controller-sa"
                }
            }
        }
    ]
}
```

b. Creare il ruolo.

```
export controller_role_name=AmazonEKS_EFS_CSI_ControllerRole
aws iam create-role \
  --role-name $controller_role_name \
  --assume-role-policy-document file://"controller-trust-policy.json"
```

3. Allega le politiche AWS gestite richieste al ruolo del controller.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonS3FilesCSIDriverPolicy \
  --role-name $controller_role_name

aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3FilesClientFullAccess \
  --role-name $controller_role_name
```

4. Crea il ruolo IAM per l'account del servizio del nodo.

a. Copia i seguenti contenuti in un file denominato `node-trust-policy.json`. Sostituisci <111122223333> con l'ID del tuo account. Sostituisci <EXAMPLED539D4633E53DE1B71EXAMPLE> e `<region-code>` con i valori restituiti nel passaggio 1.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "Federated": "arn:aws: iam::<111122223333>:oidc-provider/
oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-node-sa",
                    "oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
```

b. Crea il ruolo.

```
export node_role_name=AmazonEKS_EFS_CSI_NodeRole
aws iam create-role \
  --role-name $node_role_name \
  --assume-role-policy-document file://"node-trust-policy.json"
```

5. Allega le politiche AWS gestite richieste al ruolo del nodo.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3FilesClientFullAccess \
  --role-name $node_role_name

aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess \
  --role-name $node_role_name

aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonElasticFileSystemsUtils \
  --role-name $node_role_name
```

> **Note**
>
> La `AmazonS3ReadOnlyAccess` policy garantisce l'accesso in lettura a tutti i bucket S3. Per limitare l'accesso a bucket specifici, puoi scollegarlo e sostituirlo con una policy in linea basata su tag. Per ulteriori dettagli, consulta la documentazione della policy IAM del driver CSI GitHub di Amazon EFS.

## Fase 2: ottenere il driver CSI per Amazon EFS

Consigliamo di installare il driver Amazon EFS CSI attraverso il componente aggiuntivo di Amazon EKS. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS". Se non riesci a utilizzare il componente aggiuntivo Amazon EKS, ti consigliamo di segnalare un problema sul motivo per cui non puoi farlo all'archivio della roadmap GitHub di Containers.

> **Important**
>
> Prima di aggiungere il driver Amazon EFS come componente aggiuntivo di Amazon EKS, verificare che nel cluster non sia installata una versione autogestita del driver. In tal caso, consulta Disinstallazione del driver Amazon EFS CSI su. GitHub

In alternativa, se desideri un'installazione autogestita del driver Amazon EFS CSI, consulta Installazionesu GitHub.

## Fase 3: creare un file system Amazon S3

Per creare un file system Amazon S3, consulta Creare un file system Amazon S3 per Amazon EKS su. GitHub

## Fase 4: implementazione di un'applicazione di esempio

Puoi implementare diverse app di esempio e modificarle in base alle tue esigenze. Per ulteriori informazioni, consulta Esempi su. GitHub

# Usa l'archiviazione delle app ad alte prestazioni con Amazon FSx per Lustre

Il driver Container Storage Interface (CSI) per Amazon FSx per Lustre mette a disposizione un'interfaccia CSI che permette ai cluster Amazon EKS di gestire il ciclo di vita dei file system Amazon FSx per Lustre. Per ulteriori informazioni, consulta Amazon FSx for Lustre User Guide.

Per informazioni su come implementare il driver CSI per Amazon FSx per Lustre nel cluster Amazon EKS e come verificarne il funzionamento, consulta the section called "Implementare il driver".

## Implementa il driver FSx for Lustre

Questo argomento mostra come distribuire il driver CSI FSx for Lustre nel cluster Amazon EKS e verificarne il funzionamento. Ti consigliamo di utilizzare sempre la versione più recente del driver. Per le versioni disponibili, consulta la matrice di compatibilità delle specifiche CSI su. GitHub

> **Note**
>
> Il driver non è supportato su Fargate.

Per descrizioni dettagliate dei parametri disponibili ed esempi completi che dimostrano le funzionalità del driver, consultate il progetto driver FSx for Lustre Container Storage Interface (CSI) su. GitHub

### Prerequisiti

- Un cluster esistente.
- Il componente aggiuntivo Amazon FSx CSI Driver EKS supporta l'autenticazione tramite EKS Pod Identity o IAM Roles for Service Accounts (IRSA). Per utilizzare EKS Pod Identity, installa l'agente Pod Identity prima o dopo la distribuzione del FSx componente aggiuntivo CSI Driver. Per ulteriori informazioni, consulta the section called "Configurazione dell'agente". Per utilizzare invece IRSA, consulta. the section called "Gestore di identità OIDC IAM"
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum`apt-get, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione

e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.

- La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

## Passaggio 1: Creazione di un ruolo IAM

Il plug-in Amazon FSx CSI richiede le autorizzazioni IAM per effettuare chiamate per tuo AWS APIs conto.

> **Note**
>
> I pod avranno accesso alle autorizzazioni assegnate al ruolo IAM, a meno che non si blocchi l'accesso a IMDS. Per ulteriori informazioni, consulta the section called "Best practice".

La procedura seguente mostra come creare un ruolo IAM e allegare ad esso la policy AWS gestita.

1. Crea un ruolo IAM e collega la policy AWS gestita con il seguente comando. Sostituisci `my-cluster` con il nome del cluster che desideri utilizzare. Il comando distribuisce uno AWS CloudFormation stack che crea un ruolo IAM e vi allega la policy IAM.

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

Visualizzerai diverse righe di output quando l'account del servizio viene creato. Le ultime righe dell'output sono simili a quelle riportate di seguito.

```
[#]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
[#]  building iamserviceaccount stack "eksctl-my-cluster-addon-iamserviceaccount-
kube-system-fsx-csi-controller-sa"
[#]  deploying stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-
controller-sa"
[#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-iamserviceaccount-
kube-system-fsx-csi-controller-sa"
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

Annota il nome dello AWS CloudFormation stack che è stato distribuito. Nell'output di esempio precedente, lo stack è denominato `eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`.

Ora che hai creato il ruolo IAM del driver Amazon FSx CSI, puoi passare alla sezione successiva. Quando si implementa il componente aggiuntivo con il ruolo IAM, questo crea e viene configurato per utilizzare un account di servizio denominato `fsx-csi-controller-sa`. L'account di servizio è associato a un `clusterrole` Kubernetes a cui sono assegnate le autorizzazioni Kubernetes richieste.

## Fase 2: Installare il driver Amazon FSx CSI

Ti consigliamo di installare il driver Amazon FSx CSI tramite il componente aggiuntivo Amazon EKS per migliorare la sicurezza e ridurre la quantità di lavoro. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

> **Important**
>
> Le installazioni preesistenti di driver Amazon FSx CSI nel cluster possono causare errori di installazione dei componenti aggiuntivi. Quando tenti di installare la versione del componente aggiuntivo Amazon EKS mentre esiste un driver FSx CSI non EKS, l'installazione avrà esito

negativo a causa di conflitti di risorse. Usa il flag `OVERWRITE` durante l'installazione per risolvere il problema.

```
aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
 --resolve-conflicts OVERWRITE
```

In alternativa, se desideri un'installazione autogestita del driver Amazon FSx CSI, consulta Installation on. GitHub

## Passaggio 3: implementazione di una classe di archiviazione, una dichiarazione di volume persistente e un'app di esempio

Questa procedura utilizza il GitHub repository di driver FSx for Lustre Container Storage Interface (CSI) per utilizzare un volume Lustre con provisioning dinamico. FSx

1. Annota il gruppo di sicurezza per il cluster. Puoi vederlo nella Console di gestione AWS sezione Rete o utilizzando il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster che desideri utilizzare.

   ```
   aws eks describe-cluster --name my-cluster --query
    cluster.resourcesVpcConfig.clusterSecurityGroupId
   ```

2. Crea un gruppo di sicurezza per il tuo FSx file system Amazon in base ai criteri indicati in Amazon VPC Security Groups nella Amazon FSx for Lustre User Guide. Per il VPC, seleziona il VPC del cluster come mostrato nella sezione Reti. Per i "gruppi di sicurezza associati ai client Lustre", usa il gruppo di sicurezza del cluster. Non specificare alcun valore nelle regole in uscita per consentire Tutto il traffico.
3. Scarica il manifesto della classe di storage con il comando seguente.

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
   examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
   ```

4. Modifica la sezione dei parametri del file `storageclass.yaml`. Sostituisci ogni valore esemplificativo con i tuoi valori.

   ```
   parameters:
     subnetId: subnet-0eabfaa81fb22bcaf
     securityGroupIds: sg-068000ccf82dfba88
   ```

```
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **`subnetId`**— L'ID di sottorete in cui deve essere creato il file system Amazon FSx for Lustre. Amazon FSx for Lustre non è supportato in tutte le zone di disponibilità. Apri la console Amazon FSx for Lustre all'indirizzo per https://console.aws.amazon.com/fsx/ confermare che la sottorete che desideri utilizzare si trova in una zona di disponibilità supportata. La sottorete può includere i nodi oppure può essere una sottorete o un VPC differente:
  - Puoi verificare la presenza di sottoreti di nodi in Console di gestione AWS selezionando il gruppo di nodi nella sezione Elaborazione.
  - Se la sottorete specificata non è la stessa sottorete in cui sono presenti i nodi, è VPCs necessario essere connessi e assicurarsi di avere le porte necessarie aperte nei gruppi di sicurezza.
- **`securityGroupIds`** – l'ID del gruppo di sicurezza creato per il file system.
- **`deploymentType`** (facoltativo): il tipo di implementazione del file system. I valori validi sono `SCRATCH_1`, `SCRATCH_2`, `PERSISTENT_1` e `PERSISTENT_2`. Per ulteriori informazioni sui tipi di distribuzione, consulta Create your Amazon FSx for Lustre file system.
- altri parametri (facoltativo): per informazioni sugli altri parametri, consulta Modifica StorageClass su GitHub.

5. Crea il manifesto della classe di storage.

```
kubectl apply -f storageclass.yaml
```

Di seguito viene riportato un output di esempio:

```
storageclass.storage.k8s.io/fsx-sc created
```

6. Scaricare il manifesto della dichiarazione di volume persistente.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (Facoltativo) Modificare il file `claim.yaml`. Cambia `1200Gi` in uno dei valori di incremento riportati di seguito in base ai tuoi requisiti di archiviazione e al `deploymentType` selezionato in un passaggio precedente.

```
storage: 1200Gi
```

   - `SCRATCH_2` e `PERSISTENT` – `1.2 TiB`, `2.4 TiB` o incrementi di 2,4 TiB su 2,4 TiB.
   - `SCRATCH_1` – `1.2 TiB`, `2.4 TiB`, `3.6 TiB` o incrementi di 3,6 TiB su 3,6 TiB.

8. Creare la dichiarazione di volume persistente.

```
kubectl apply -f claim.yaml
```

   Di seguito viene riportato un output di esempio:

```
persistentvolumeclaim/fsx-claim created
```

9. Verificare che il file system sia stato sottoposto a provisioning.

```
kubectl describe pvc
```

   Di seguito viene riportato un output di esempio:

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

   > **Note**
   >
   > `Status` può apparire `Pending` per 5-10 minuti, prima di passare a `Bound`. Non continuare con il passo successivo fino a quando `Status` non è `Bound`. Se nel campo `Status` viene visualizzato `Pending` per più di 10 minuti, utilizza i messaggi di avviso negli `Events` come riferimento per risolvere eventuali problemi.

10. Implementare un'applicazione di esempio.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-
driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. Verificare che l'applicazione di esempio sia in esecuzione.

```
kubectl get pods
```

Di seguito viene riportato un output di esempio:

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. Verifica che l'applicazione abbia montato correttamente il file system.

```
kubectl exec -ti fsx-app -- df -h
```

Di seguito viene riportato un output di esempio.

```
Filesystem                  Size  Used Avail Use% Mounted on
overlay                      80G  4.0G   77G   5% /
tmpfs                        64M     0   64M   0% /dev
tmpfs                       3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01     1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1               80G  4.0G   77G   5% /etc/hosts
shm                          64M     0   64M   0% /dev/shm
tmpfs                       6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                       3.8G     0  3.8G   0% /proc/acpi
tmpfs                       3.8G     0  3.8G   0% /sys/firmware
```

13. Verifica che i dati siano stati scritti nel file system FSx for Lustre dall'app di esempio.

```
kubectl exec -it fsx-app -- ls /data
```

Di seguito viene riportato un output di esempio.

```
out.txt
```

Questo output esemplificativo mostra che l'app di esempio ha eseguito correttamente la scrittura del file `out.txt` nel file system.

> **Note**
>
> Prima di eliminare il cluster, assicuratevi di eliminare il file system FSx for Lustre. Per ulteriori informazioni, consultate Clean up resources nella Guida FSx per l'utente di for Lustre.

### Ottimizzazione delle prestazioni per for FSx Lustre

Quando si utilizza FSx per Lustre con Amazon EKS, è possibile ottimizzare le prestazioni applicando le ottimizzazioni Lustre durante l'inizializzazione del nodo. L'approccio consigliato consiste nell'utilizzare i dati utente del modello di avvio per garantire una configurazione coerente su tutti i nodi.

Queste ottimizzazioni includono:

- Ottimizzazioni di rete e RPC
- Gestione del modulo Lustre
- Ottimizzazioni LRU (Lock Resource Unit)
- Impostazioni di controllo della cache del client
- Controlli RPC per OST e MDC

Per istruzioni dettagliate sull'implementazione di queste ottimizzazioni delle prestazioni:

- Per ottimizzare le prestazioni per i nodi standard (non EFA), consulta the section called "Ottimizza (non EFA)" per uno script completo che può essere aggiunto ai dati utente del modello di avvio.
- Per ottimizzare le prestazioni per i nodi abilitati all'EFA, consulta the section called "Ottimizza (EFA)".

## Ottimizza le prestazioni di Amazon FSx per Lustre sui nodi (EFA)

In questa sezione viene descritto come configurare l'ottimizzazione di Elastic Fabric Adapter (EFA) con Amazon EKS e Amazon FSx per Lustre.

> **Note**
>
> - Per informazioni sulla creazione e l'implementazione del driver CSI per FSx per Lustre, consultare the section called "Implementare il driver".
> - Per ottimizzare i nodi standard senza EFA, consultare the section called "Ottimizza (non EFA)".

## Passaggio 1. Crea un cluster EKS

Crea un cluster utilizzando il file di configurazione fornito:

```
# Create cluster using efa-cluster.yaml
eksctl create cluster -f efa-cluster.yaml
```

Esempio`efa-cluster.yaml`:

```
#efa-cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: csi-fsx
  region: us-east-1
  version: "1.30"

iam:
  withOIDC: true

availabilityZones: ["us-east-1a", "us-east-1d"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1b"]
    volumeSize: 300
    privateNetworking: true
```

```
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print $1}')"
        efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//g')
        min_efa_version="2.12.1"

        if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
            sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
            tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer
            echo "Installing EFA driver"
            sudo apt-get update && apt-get upgrade -y
            sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms
            sudo ./efa_installer.sh -y
            modinfo efa
        else
            echo "Using EFA driver version $efa_version"
        fi

        echo "Installing Lustre client"
        sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-
ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-
key.gpg > /dev/null
        sudo echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /etc/apt/
sources.list.d/fsxlustreclientrepo.list
        sudo apt update | tail
        sudo apt install -y lustre-client-modules-$(uname -r) amazon-ec2-utils | tail
        modinfo lustre

        echo "Loading Lustre/EFA modules..."
        sudo /sbin/modprobe lnet
        sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
        sudo /sbin/modprobe ksocklnd
        sudo lnetctl lnet configure

        echo "Configuring TCP interface..."
        sudo lnetctl net del --net tcp 2> /dev/null
        sudo lnetctl net add --net tcp --if $eth_intf
```

```
        # For P5 instance type which supports 32 network cards,
        # by default add 8 EFA interfaces selecting every 4th device (1 per PCI bus)
        echo "Configuring EFA interface(s)..."
        instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
        num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
        echo "Found $num_efa_devices available EFA device(s)"

        if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
 "p5e.48xlarge" ]]; then
           for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
               sudo lnetctl net add --net efa --if $intf --peer-credits 32
          done
        else
        # Other instances: Configure 2 EFA interfaces by default if the instance
 supports multiple network cards,
        # or 1 interface for single network card instances
        # Can be modified to add more interfaces if instance type supports it
            sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | head -
n1) --peer-credits 32
            if [[ $num_efa_devices -gt 1 ]]; then
               sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | tail
 -n1) --peer-credits 32
            fi
        fi

        echo "Setting discovery and UDSP rule"
        sudo lnetctl set discovery 1
        sudo lnetctl udsp add --src efa --priority 0
        sudo /sbin/modprobe lustre

        sudo lnetctl net show
        echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
```

## Passaggio 2. Creazione di un gruppo di nodi

Crea un gruppo di EFA-enabled nodi:

```
# Create node group using efa-ng.yaml
eksctl create nodegroup -f efa-ng.yaml
```

**Important**

Modifica questi valori per il tuo ambiente nella sezione`# 5. Mount FSx filesystem`.

```
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

Esempio`efa-ng.yaml`:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: final-efa
  region: us-east-1

managedNodeGroups:
  - name: ng-1
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1a"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        exec 1> >(logger -s -t $(basename $0)) 2>&1


 ############################################################################################
        #                                      Configuration Section
         #

 ############################################################################################

        # File System Configuration
```

```
        FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
        MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
        MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.

        # Lustre Tuning Parameters
        LUSTRE_LRU_MAX_AGE=600000
        LUSTRE_MAX_CACHED_MB=64
        LUSTRE_OST_MAX_RPC=32
        LUSTRE_MDC_MAX_RPC=64
        LUSTRE_MDC_MOD_RPC=50

        # File paths
        FUNCTIONS_SCRIPT="/usr/local/bin/lustre_functions.sh"
        TUNINGS_SCRIPT="/usr/local/bin/apply_lustre_tunings.sh"
        SERVICE_FILE="/etc/systemd/system/lustre-tunings.service"

        #EFA
        MIN_EFA_VERSION="2.12.1"

        # Function to check if a command was successful
        check_success() {
            if [ $? -eq 0 ]; then
                echo "SUCCESS: $1"
            else
                echo "FAILED: $1"
                return 1
            fi
        }

        echo "********Starting FSx for Lustre configuration********"

        # 1. Install Lustre client
        if grep -q '^ID=ubuntu' /etc/os-release; then
            echo "Detected Ubuntu, proceeding with Lustre setup..."
            # Add Lustre repository
            sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/
fsx-ubuntu-public-key.asc | sudo gpg --dearmor | sudo tee /usr/share/keyrings/fsx-
ubuntu-public-key.gpg > /dev/null

            echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" | sudo tee /etc/
apt/sources.list.d/fsxlustreclientrepo.list

            sudo apt-get update
```

```
            sudo apt-get install -y lustre-client-modules-$(uname -r)
            sudo apt-get install -y lustre-client
        else
            echo "Not Ubuntu, exiting"
            exit 1
        fi

        check_success "Install Lustre client"

        # Ensure Lustre tools are in the PATH
        export PATH=$PATH:/usr/sbin

        # 2. Apply network and RPC tunings
        echo "********Applying network and RPC tunings********"
        if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/
modprobe.conf; then
            echo "options ptlrpc ptlrpcd_per_cpt_max=64" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ptlrpcd_per_cpt_max"
        else
            echo "ptlrpcd_per_cpt_max already set in modprobe.conf"
        fi

        if ! grep -q "options ksocklnd credits" /etc/modprobe.d/modprobe.conf; then
            echo "options ksocklnd credits=2560" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ksocklnd credits"
        else
            echo "ksocklnd credits already set in modprobe.conf"
        fi

        # 3. Load Lustre modules
        manage_lustre_modules() {
            echo "Checking for existing Lustre modules..."
            if lsmod | grep -q lustre; then
                echo "Existing Lustre modules found."

                # Check for mounted Lustre filesystems
                echo "Checking for mounted Lustre filesystems..."
                if mount | grep -q "type lustre"; then
                    echo "Found mounted Lustre filesystems. Attempting to unmount..."
                    mounted_fs=$(mount | grep "type lustre" | awk '{print $3}')
                    for fs in $mounted_fs; do
                        echo "Unmounting $fs"
```

```
                    sudo umount $fs
                    check_success "Unmount filesystem $fs"
                done
            else
                echo "No Lustre filesystems mounted."
            fi

            # After unmounting, try to remove modules
            echo "Attempting to remove Lustre modules..."
            sudo lustre_rmmod
            if [ $? -eq 0 ]; then
                echo "SUCCESS: Removed existing Lustre modules"
            else
                echo "WARNING: Could not remove Lustre modules. They may still be
in use."
                echo "Please check for any remaining Lustre processes or mounts."
                return 1
            fi
        else
            echo "No existing Lustre modules found."
        fi

        echo "Loading Lustre modules..."
        sudo modprobe lustre
        check_success "Load Lustre modules" || exit 1

        echo "Checking loaded Lustre modules:"
        lsmod | grep lustre
    }

    # Managing Lustre kernel modules
    echo "********Managing Lustre kernel modules********"
    manage_lustre_modules

    # 4. Initializing Lustre networking
    echo "********Initializing Lustre networking********"
    sudo lctl network up
    check_success "Initialize Lustre networking" || exit 1

    # 4.5 EFA Setup and Configuration
    setup_efa() {
        echo "********Starting EFA Setup********"

        # Get interface and version information
```

```
            eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print
 $1}')"
            efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//
g')
            min_efa_version=$MIN_EFA_VERSION

            # Install or verify EFA driver
            if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
                echo "Installing EFA driver..."
                sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
                tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer

                # Install dependencies
                sudo apt-get update && apt-get upgrade -y
                sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms

                # Install EFA
                sudo ./efa_installer.sh -y
                modinfo efa
            else
                echo "Using existing EFA driver version $efa_version"
            fi
        }

        configure_efa_network() {
            echo "********Configuring EFA Network********"

            # Load required modules
            echo "Loading network modules..."
            sudo /sbin/modprobe lnet
            sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
            sudo /sbin/modprobe ksocklnd

            # Initialize LNet
            echo "Initializing LNet..."
            sudo lnetctl lnet configure

            # Configure TCP interface
            echo "Configuring TCP interface..."
            sudo lnetctl net del --net tcp 2> /dev/null
            sudo lnetctl net add --net tcp --if $eth_intf
```

```
            # For P5 instance type which supports 32 network cards,
            # by default add 8 EFA interfaces selecting every 4th device (1 per PCI
bus)
            echo "Configuring EFA interface(s)..."
            instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
            num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
            echo "Found $num_efa_devices available EFA device(s)"

            if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
"p5e.48xlarge" ]]; then
                # P5 instance configuration
                for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
                    sudo lnetctl net add --net efa --if $intf --peer-credits 32
                done
            else
                # Standard configuration
                # Other instances: Configure 2 EFA interfaces by default if the
instance supports multiple network cards,
                # or 1 interface for single network card instances
                # Can be modified to add more interfaces if instance type supports it
                sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
head -n1) --peer-credits 32
                if [[ $num_efa_devices -gt 1 ]]; then
                    sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
tail -n1) --peer-credits 32
                fi
            fi

            # Configure discovery and UDSP
            echo "Setting up discovery and UDSP rules..."
            sudo lnetctl set discovery 1
            sudo lnetctl udsp add --src efa --priority 0
            sudo /sbin/modprobe lustre

            # Verify configuration
            echo "Verifying EFA network configuration..."
            sudo lnetctl net show
            echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
        }

        # Main execution
        setup_efa
        configure_efa_network
```

```
        # 5. Mount FSx filesystem
        if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
            echo "********Creating mount point********"
            sudo mkdir -p $MOUNT_POINT
            check_success "Create mount point"

            echo "Mounting FSx filesystem..."
            sudo mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
            check_success "Mount FSx filesystem"
        else
            echo "Skipping FSx mount as DNS or mount name is not provided"
        fi

        # 6. Applying Lustre performance tunings
        echo "********Applying Lustre performance tunings********"

        # Get number of CPUs
        NUM_CPUS=$(nproc)

        # Calculate LRU size (100 * number of CPUs)
        LRU_SIZE=$((100 * NUM_CPUS))

        #Apply LRU tunings
        echo "Apply LRU tunings"
        sudo lctl set_param ldlm.namespaces.*.lru_max_age=${LUSTRE_LRU_MAX_AGE}
        check_success "Set lru_max_age"
        sudo lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
        check_success "Set lru_size"

        # Client Cache Control
        sudo lctl set_param llite.*.max_cached_mb=${LUSTRE_MAX_CACHED_MB}
        check_success "Set max_cached_mb"

        # RPC Controls
        sudo lctl set_param osc.*OST*.max_rpcs_in_flight=${LUSTRE_OST_MAX_RPC}
        check_success "Set OST max_rpcs_in_flight"

        sudo lctl set_param mdc.*.max_rpcs_in_flight=${LUSTRE_MDC_MAX_RPC}
        check_success "Set MDC max_rpcs_in_flight"

        sudo lctl set_param mdc.*.max_mod_rpcs_in_flight=${LUSTRE_MDC_MOD_RPC}
        check_success "Set MDC max_mod_rpcs_in_flight"
```

```
        # 7. Verify all tunings
        echo "********Verifying all tunings********"

        # Function to verify parameter value
        verify_param() {
            local param=$1
            local expected=$2
            local actual=$3

            if [ "$actual" == "$expected" ]; then
                echo "SUCCESS: $param is correctly set to $expected"
            else
                echo "WARNING: $param is set to $actual (expected $expected)"
            fi
        }

        echo "Verifying all parameters:"

        # LRU tunings
        actual_lru_max_age=$(lctl get_param -n ldlm.namespaces.*.lru_max_age | head -1)
        verify_param "lru_max_age" "600000" "$actual_lru_max_age"

        actual_lru_size=$(lctl get_param -n ldlm.namespaces.*.lru_size | head -1)
        verify_param "lru_size" "$LRU_SIZE" "$actual_lru_size"

        # Client Cache
        actual_max_cached_mb=$(lctl get_param -n llite.*.max_cached_mb | grep
"max_cached_mb:" | awk '{print $2}')
        verify_param "max_cached_mb" "64" "$actual_max_cached_mb"

        # RPC Controls
        actual_ost_rpcs=$(lctl get_param -n osc.*OST*.max_rpcs_in_flight | head -1)
        verify_param "OST max_rpcs_in_flight" "32" "$actual_ost_rpcs"

        actual_mdc_rpcs=$(lctl get_param -n mdc.*.max_rpcs_in_flight | head -1)
        verify_param "MDC max_rpcs_in_flight" "64" "$actual_mdc_rpcs"

        actual_mdc_mod_rpcs=$(lctl get_param -n mdc.*.max_mod_rpcs_in_flight | head -1)
        verify_param "MDC max_mod_rpcs_in_flight" "50" "$actual_mdc_mod_rpcs"

        # Network and RPC configurations from modprobe.conf
        actual_ptlrpc=$(grep "ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf
| awk '{print $3}')
        verify_param "ptlrpcd_per_cpt_max" "ptlrpcd_per_cpt_max=64" "$actual_ptlrpc"
```

```
        actual_ksocklnd=$(grep "ksocklnd credits" /etc/modprobe.d/modprobe.conf | awk
'{print $3}')
        verify_param "ksocklnd credits" "credits=2560" "$actual_ksocklnd"

        # 8. Setup persistence
        setup_persistence() {
            # Create functions file
            cat << EOF > $FUNCTIONS_SCRIPT
        #!/bin/bash

        apply_lustre_tunings() {
            local NUM_CPUS=\$(nproc)
            local LRU_SIZE=\$((100 * NUM_CPUS))

            echo "Applying Lustre performance tunings..."
            lctl set_param ldlm.namespaces.*.lru_max_age=$LUSTRE_LRU_MAX_AGE
            lctl set_param ldlm.namespaces.*.lru_size=\$LRU_SIZE
            lctl set_param llite.*.max_cached_mb=$LUSTRE_MAX_CACHED_MB
            lctl set_param osc.*OST*.max_rpcs_in_flight=$LUSTRE_OST_MAX_RPC
            lctl set_param mdc.*.max_rpcs_in_flight=$LUSTRE_MDC_MAX_RPC
            lctl set_param mdc.*.max_mod_rpcs_in_flight=$LUSTRE_MDC_MOD_RPC
        }
        EOF

            # Create tuning script
            cat << EOF > $TUNINGS_SCRIPT
        #!/bin/bash
        exec 1> >(logger -s -t \$(basename \$0)) 2>&1

        source $FUNCTIONS_SCRIPT

        # Function to check if Lustre is mounted
        is_lustre_mounted() {
            mount | grep -q "type lustre"
        }

        # Function to mount Lustre
        mount_lustre() {
            echo "Mounting Lustre filesystem..."
            mkdir -p $MOUNT_POINT
            mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} $MOUNT_POINT
            return \$?
        }
```

```
# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi

# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done

echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF

# Create systemd service

# Create systemd directory if it doesn't exist
sudo mkdir -p /etc/systemd/system/

    # Create service file directly for Ubuntu
    cat << EOF > $SERVICE_FILE
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target

[Service]
Type=oneshot
ExecStart=/bin/bash -c 'source $FUNCTIONS_SCRIPT && $TUNINGS_SCRIPT'
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
EOF
```

```
            # Make scripts executable and enable service
            sudo chmod +x $FUNCTIONS_SCRIPT
            sudo chmod +x $TUNINGS_SCRIPT
            systemctl enable lustre-tunings.service
            systemctl start lustre-tunings.service
        }

        echo "********Setting up persistent tuning********"
        setup_persistence

        echo "FSx for Lustre configuration completed."
```

## (Facoltativo) Passaggio 3. Verifica della configurazione EFA

SSH nel nodo:

```
# Get instance ID from EKS console or {aws} CLI
ssh -i /path/to/your-key.pem ec2-user@<node-internal-ip>
```

Verifica della configurazione EFA:

```
sudo lnetctl net show
```

Controlla i registri di configurazione:

```
sudo cat /var/log/cloud-init-output.log
```

Ecco un esempio di output previsto per `lnetctl net show`:

```
net:
    - net type: tcp
      ...
    - net type: efa
      local NI(s):
        - nid: xxx.xxx.xxx.xxx@efa
          status: up
```

## Esempio di implementazioni

### a. Creazione di claim.yaml

```
#claim.yaml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fsx-claim-efa
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 4800Gi
  volumeName: fsx-pv
```

Applica la richiesta:

```
kubectl apply -f claim.yaml
```

### b. Creazione di pv.yaml

Aggiorna il`<replaceable-placeholders>`:

```
#pv.yaml

apiVersion: v1
kind: PersistentVolume
metadata:
  name: fsx-pv
spec:
  capacity:
    storage: 4800Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  mountOptions:
    - flock
  persistentVolumeReclaimPolicy: Recycle
```

```
  csi:
    driver: fsx.csi.aws.com
    volumeHandle: fs-<1234567890abcdef0>
    volumeAttributes:
      dnsname: fs-<1234567890abcdef0>.fsx.us-east-1.amazonaws.com
      mountname: <abcdef01>
```

Applica il volume persistente:

```
kubectl apply -f pv.yaml
```

c. Creazione di pod.yaml

```
#pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: fsx-efa-app
spec:
  containers:
  - name: app
    image: amazonlinux:2
    command: ["/bin/sh"]
    args: ["-c", "while true; do dd if=/dev/urandom bs=100M count=20 > data/test_file;
 sleep 10; done"]
    resources:
      requests:
        vpc.amazonaws.com/efa: 1
      limits:
        vpc.amazonaws.com/efa: 1
    volumeMounts:
    - name: persistent-storage
      mountPath: /data
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: fsx-claim-efa
```

Applica il pod:

```
kubectl apply -f pod.yaml
```

### Comandi di verifica aggiuntivi

Verifica i montaggi del pod e scrive sul filesystem:

```
kubectl exec -ti fsx-efa-app -- df -h | grep data
# Expected output:
# <192.0.2.0>@tcp:/<abcdef01>  4.5T  1.2G  4.5T   1% /data

kubectl exec -ti fsx-efa-app -- ls /data
# Expected output:
# test_file
```

SSH sul nodo per verificare che il traffico stia passando attraverso EFA:

```
sudo lnetctl net show -v
```

L'output previsto mostrerà le interfacce EFA con le statistiche sul traffico.

### Informazioni correlate

- the section called "Implementare il driver"
- the section called "Ottimizza (non EFA)"
- Prestazioni Amazon FSx per Lustre
- Elastic Fabric Adapter

## Ottimizza le prestazioni di Amazon FSx per Lustre sui nodi (non EFA)

Puoi ottimizzare le prestazioni di Amazon FSx per Lustre applicando i parametri di ottimizzazione durante l'inizializzazione del nodo utilizzando i dati utente del modello di avvio.

> **Note**
>
> - Per informazioni sulla creazione e l'implementazione del driver CSI per FSx per Lustre, consultare the section called "Implementare il driver".
> - Per ottimizzare le prestazioni con EFA-enabled i nodi, vederethe section called "Ottimizza (EFA)".

## Perché utilizzare i dati utente del modello di avvio?

- Applica automaticamente le ottimizzazioni durante l'inizializzazione del nodo.
- Garantisce una configurazione coerente su tutti i nodi.
- Elimina la necessità di configurare manualmente il nodo.

## Panoramica degli script di esempio

Lo script di esempio definito in questo argomento esegue le seguenti operazioni:

**`# 1. Installa il client Lustre`**

- Rileva automaticamente la versione del tuo sistema operativo Amazon Linux (AL).
- Installa il pacchetto client Lustre appropriato.

**`# 2. Applica le ottimizzazioni di rete e RPC`**

- Imposta `ptlrpcd_per_cpt_max=64` per l'elaborazione RPC parallela.
- Configura `ksocklnd credits=2560` per ottimizzare i buffer di rete.

**`# 3. Moduli Load Lustre`**

- Rimuove in modo sicuro i moduli Lustre esistenti, se presenti.
- Gestisce lo smontaggio dei filesystem esistenti.
- Carica nuovi moduli Lustre.

**`# 4. Inizializzazione della rete Lustre`**

- Inizializza la configurazione di rete Lustre.
- Imposta i parametri di rete richiesti.

**`# 5. Montare il file system FSx`**

- È necessario modificare i valori per il tuo ambiente in questa sezione.

**# 6. Applica le regolazioni**

- Ottimizzazioni LRU (Lock Resource Unit):
  - `lru_max_age=600000`
  - `lru_size` calcolato in base al numero di CPU
- Controllo della cache del client: `max_cached_mb=64`
- Controlli RPC:
  - OST `max_rpcs_in_flight=32`
  - MDC `max_rpcs_in_flight=64`
  - MDC `max_mod_rpcs_in_flight=50`

**# 7. Verifica le regolazioni**

- Verifica tutte le ottimizzazioni applicate.
- Segnala il successo o l'avviso per ogni parametro.

**# 8. Persistenza dell'installazione**

- È necessario modificare i valori per il tuo ambiente anche in questa sezione.
- Rileva automaticamente la versione del sistema operativo (AL2023) per determinare `Systemd` quale servizio applicare.
- Il sistema si avvia.
- `Systemd` avvia il servizio `lustre-tunings` (a causa di `WantedBy=multi-user.target`).
- Il servizio esegue `apply_lustre_tunings.sh` che:
  - Verifica se il filesystem è montato.
  - Monta il filesystem se non è montato.
  - Attende che il montaggio venga eseguito con successo (fino a cinque minuti).
  - Applica i parametri di ottimizzazione dopo il montaggio riuscito.
- Le impostazioni rimangono attive fino al riavvio.
- Il servizio si chiude dopo il completamento dello script.
  - Systemd contrassegna il servizio come "attivo (terminato)".
- Il processo si ripete al successivo riavvio.

## Creazione di un modello di avvio

1. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Scegli Modelli di avvio.
3. Scegli Crea modello di avvio.
4. In Dettagli avanzati, individua la sezione Dati utente.
5. Incolla lo script qui di seguito, aggiornando tutto ciò che è necessario.

> **Important**
>
> Regola questi valori per il tuo ambiente sia nella sezione `# 5. Mount FSx filesystem` che nella funzione `setup_persistence()` di `apply_lustre_tunings.sh` nella sezione `# 8. Setup persistence`:
>
> ```
> FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
> MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
> MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
> ```

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Function definitions
check_success() {
    if [ $? -eq 0 ]; then
        echo "SUCCESS: $1"
    else
        echo "FAILED: $1"
        return 1
    fi
}
apply_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age=600000"
```

```
        "ldlm.namespaces.*.lru_size=$LRU_SIZE"
        "llite.*.max_cached_mb=64"
        "osc.*OST*.max_rpcs_in_flight=32"
        "mdc.*.max_rpcs_in_flight=64"
        "mdc.*.max_mod_rpcs_in_flight=50"
    )
    for param in "${params[@]}"; do
        lctl set_param $param
        check_success "Set ${param%%=*}"
    done
}
verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
        echo "WARNING: $param is set to $actual (expected $expected)"
    fi
}
verify_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age:600000"
        "ldlm.namespaces.*.lru_size:$LRU_SIZE"
        "llite.*.max_cached_mb:64"
        "osc.*OST*.max_rpcs_in_flight:32"
        "mdc.*.max_rpcs_in_flight:64"
        "mdc.*.max_mod_rpcs_in_flight:50"
    )
    echo "Verifying all parameters:"
    for param in "${params[@]}"; do
        name="${param%%:*}"
        expected="${param#*:}"
        actual=$(lctl get_param -n $name | head -1)
        verify_param "${name##*.}" "$expected" "$actual"
    done
}
setup_persistence() {
    # Create functions file
    cat << 'EOF' > /usr/local/bin/lustre_functions.sh
```

```
#!/bin/bash
apply_lustre_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))

    echo "Applying Lustre performance tunings..."
    lctl set_param ldlm.namespaces.*.lru_max_age=600000
    lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
    lctl set_param llite.*.max_cached_mb=64
    lctl set_param osc.*OST*.max_rpcs_in_flight=32
    lctl set_param mdc.*.max_rpcs_in_flight=64
    lctl set_param mdc.*.max_mod_rpcs_in_flight=50
}
EOF
    # Create tuning script
    cat << 'EOF' > /usr/local/bin/apply_lustre_tunings.sh
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Source the functions
source /usr/local/bin/lustre_functions.sh
# FSx details
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
# Function to check if Lustre is mounted
is_lustre_mounted() {
    mount | grep -q "type lustre"
}
# Function to mount Lustre
mount_lustre() {
    echo "Mounting Lustre filesystem..."
    mkdir -p ${MOUNT_POINT}
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    return $?
}
# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi
# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
```

```
        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done
echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF
    # Create systemd service
    cat << 'EOF' > /etc/systemd/system/lustre-tunings.service
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target
StartLimitIntervalSec=0
[Service]
Type=oneshot
ExecStart=/usr/local/bin/apply_lustre_tunings.sh
RemainAfterExit=yes
Restart=on-failure
RestartSec=30
[Install]
WantedBy=multi-user.target
EOF
    chmod +x /usr/local/bin/lustre_functions.sh
    chmod +x /usr/local/bin/apply_lustre_tunings.sh
    systemctl enable lustre-tunings.service
    systemctl start lustre-tunings.service
}
echo "Starting FSx for Lustre configuration..."
# 1. Install Lustre client
if grep -q 'VERSION="2"' /etc/os-release; then
    amazon-linux-extras install -y lustre
elif grep -q 'VERSION="2023"' /etc/os-release; then
    dnf install -y lustre-client
fi
check_success "Install Lustre client"
# 2. Apply network and RPC tunings
export PATH=$PATH:/usr/sbin
echo "Applying network and RPC tunings..."
if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf; then
    echo "options ptlrpc ptlrpcd_per_cpt_max=64" | tee -a /etc/modprobe.d/
modprobe.conf
```

```
    echo "options ksocklnd credits=2560" | tee -a /etc/modprobe.d/modprobe.conf
fi
# 3. Load Lustre modules
modprobe lustre
check_success "Load Lustre modules" || exit 1
# 4. Lustre Network Initialization
lctl network up
check_success "Initialize Lustre networking" || exit 1
# 5. Mount FSx filesystem
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
    mkdir -p $MOUNT_POINT
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    check_success "Mount FSx filesystem"
fi
# 6. Apply tunings
apply_tunings
# 7. Verify tunings
verify_tunings
# 8. Setup persistence
setup_persistence
echo "FSx for Lustre configuration completed."
--==MYBOUNDARY==--
```

6. Quando crei gruppi di nodi Amazon EKS, seleziona questo modello di avvio. Per ulteriori informazioni, consulta the section called "Crea".

### Informazioni correlate

- the section called "Implementare il driver"
- the section called "Ottimizza (EFA)"
- Prestazioni Amazon FSx per Lustre

## Usa lo storage delle app ad alte prestazioni con FSx for ONTAP NetApp

NetApp Trident fornisce un'orchestrazione dinamica dello storage utilizzando un driver conforme a Container Storage Interface (CSI). Ciò consente ai cluster Amazon EKS di gestire il ciclo di vita dei

volumi persistenti (PV) supportati da Amazon FSx per i file system ONTAP. NetApp Tieni presente che il driver CSI Amazon FSx for NetApp ONTAP non è compatibile con Amazon EKS Hybrid Nodes. Per iniziare, consulta Utilizzare Trident con Amazon FSx for NetApp ONTAP nella documentazione di Trident. NetApp

Amazon FSx for NetApp ONTAP è un servizio di storage che consente di avviare ed eseguire file system ONTAP completamente gestiti nel cloud. ONTAP è NetApp la tecnologia di file system che fornisce una serie ampiamente adottata di funzionalità di accesso e gestione dei dati. FSx for ONTAP offre le funzionalità, le prestazioni e le API dei NetApp file system locali con l'agilità, la scalabilità e la semplicità di un servizio completamente gestito. AWS Per ulteriori informazioni, consulta la Guida per l'utente di FSx per ONTAP.

**Important**

Se utilizzi Amazon FSx for NetApp ONTAP insieme al driver Amazon EBS CSI per il provisioning di volumi EBS, devi specificare di non utilizzare dispositivi EBS nel file. `multipath.conf` Per i metodi supportati, consulta Configuration File Blacklist. Ecco un esempio.

```
defaults {
       user_friendly_names yes
       find_multipaths no
     }
     blacklist {
       device {
         vendor "NVME"
         product "Amazon Elastic Block Store"
       }
     }
```

## Usa l'archiviazione di dati con Amazon FSx per OpenZFS

Amazon FSx for OpenZFS è un servizio di archiviazione di file completamente gestito che semplifica lo spostamento dei dati da ZFS o altri file server locali. AWS Linux-based Puoi farlo senza modificare il codice dell'applicazione o la modalità di gestione dei dati. Offre un'archiviazione di file altamente affidabile, scalabile, efficiente e ricca di funzionalità basata sul file system OpenZFS open source. Combina queste funzionalità con l'agilità, la scalabilità e la semplicità di un servizio completamente gestito. AWS Per ulteriori informazioni, consulta la Guida per l'utente di Amazon FSx per OpenZFS.

Il driver FSx for OpenZFS Container Storage Interface (CSI) fornisce un'interfaccia CSI che consente ai cluster Amazon EKS di gestire il ciclo di vita dei volumi FSx per OpenZFS. Tieni presente che il Driver CSI di Amazon FSx per OpenZFS non è compatibile con Amazon EKS Hybrid Nodes. Per distribuire il driver CSI FSx for OpenZFS nel tuo cluster Amazon EKS, consulta aws-fsx-openzfs-csi-driver on. https://github.com/kubernetes-sigs/aws-fsx-openzfs-csi-driver GitHub

# Riduci al minimo la latenza con Amazon File Cache

Amazon File Cache è una cache ad alta velocità completamente gestita utilizzata per elaborare AWS i dati dei file, indipendentemente da dove sono archiviati i dati. Amazon File Cache carica automaticamente i dati nella cache quando vi si accede per la prima volta e li rilascia quando non vengono utilizzati. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon File Cache.

Il driver Amazon File Cache Container Storage Interface (CSI) fornisce un'interfaccia CSI che consente ai cluster Amazon EKS di gestire il ciclo di vita delle cache di file Amazon. Tieni presente che il driver Amazon File Cache CSI non è compatibile con Amazon EKS Hybrid Nodes. Per distribuire il driver CSI Amazon File Cache nel tuo cluster Amazon EKS, consulta aws-file-cache-csi-driver on. https://github.com/kubernetes-sigs/aws-file-cache-csi-driver GitHub

# Accedere agli oggetti Amazon S3 con il driver CSI Mountpoint per Amazon S3

Con il driver dell'interfaccia di archiviazione del container (CSI) di Mountpoint per Amazon S3, le applicazioni Kubernetes possono accedere agli oggetti Amazon S3 tramite un'interfaccia di file system, raggiungendo una throughput aggregato elevato senza modificare alcun codice applicativo. Basato su Mountpoint per Amazon S3, il driver CSI presenta un bucket Amazon S3 come volume, a cui è possibile accedere tramite i container in Amazon EKS e i cluster Kubernetes autogestiti.

## Considerazioni

- Il driver CSI di Mountpoint per Amazon S3 non è compatibile con le immagini di container basate su Windows.
- Il driver CSI di Mountpoint per Amazon S3 non è attualmente compatibile con Amazon EKS Hybrid Nodes.
- Il driver CSI di Mountpoint per Amazon S3 non supporta AWS Fargate. Tuttavia, sono supportati i container in esecuzione in Amazon EC2 (con Amazon EKS o un'installazione Kubernetes personalizzata).

- Il driver CSI di Mountpoint per Amazon S3 supporta solo il provisioning statico. Non è supportato il provisioning dinamico o la creazione di nuovi bucket.

> **Note**
>
> Il provisioning statico si riferisce all'utilizzo di un bucket Amazon S3 esistente specificato come `bucketName` in `volumeAttributes` nell'oggetto `PersistentVolume`. Per ulteriori informazioni, consulta Provisioning statico su GitHub.

- I volumi montati con il driver CSI di Mountpoint per Amazon S3 non supportano tutte le funzionalità del file system POSIX. Per i dettagli sul comportamento del file system, consulta Comportamento del file system di Mountpoint per Amazon S3 su GitHub.

Per i dettagli sull'implementazione del driver, consulta the section called "Implementazione del driver". Per informazioni dettagliate sulla rimozione del driver, consulta the section called "Rimozione del driver".

## Implementazione del driver Mountpoint per Amazon S3

Con il driver dell'interfaccia di archiviazione del container (CSI) di Mountpoint per Amazon S3, le applicazioni Kubernetes possono accedere agli oggetti Amazon S3 tramite un'interfaccia di file system, raggiungendo una throughput aggregato elevato senza modificare alcun codice applicativo.

Questa procedura illustra come implementare il driver Amazon EKS CSI di Mountpoint per Amazon S3. Prima di procedere, rivedi le considerazioni.

### Prerequisiti

- Un provider OpenID Connect (OIDC) di AWS Identity and Access Management (IAM) esistente per il tuo cluster. Per determinare se si dispone già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Versione `2.12.3` o successiva della AWS CLI installata e configurata sul dispositivo o. AWS CloudShell
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

## Fase 1: Creazione di una policy IAM

Il driver CSI di Mountpoint per Amazon S3 richiede le autorizzazioni Amazon S3 per interagire con il file system. In questa sezione viene descritto come creare una policy IAM che conceda le autorizzazioni necessarie.

La seguente policy esemplificativa segue i suggerimenti delle autorizzazioni IAM per Mountpoint. In alternativa, puoi utilizzare la policy AWS gestita AmazonS3FullAccess, ma questa policy gestita concede più autorizzazioni di quelle necessarie per Mountpoint.

Per ulteriori informazioni sulle autorizzazioni consigliate per Mountpoint, consulta Autorizzazioni IAM di Mountpoint on. GitHub

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Policy.
3. Nella pagina Policy, scegli Crea policy.
4. Per Editor di policy, scegli JSON.
5. In Editor di policy, copia e incolla quanto segue:

> **Important**
>
> Sostituisci `amzn-s3-demo-bucket1` con il nome del bucket Amazon S3.

```
{
   "Version":"2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
    ]
}
```

I bucket di directory, introdotti con la classe di archiviazione Amazon S3 Express One Zone, utilizzano un meccanismo di autenticazione diverso dai bucket per uso generico. Invece di utilizzare le azioni `s3:*`, devi utilizzare l'azione `s3express:CreateSession`. Per ulteriori informazioni sui bucket, consulta Bucket di directory nella Guida per l'utente di Amazon S3.

Di seguito è riportato un esempio di policy dei privilegi minimi da utilizzare per un bucket di directory.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket1--usw2-az1--x-s3"
        }
    ]
}
```

6. Scegli Avanti.
7. Nella pagina Verifica e crea, assegna un nome alla policy. Questa procedura guidata di esempio utilizza il nome `AmazonS3CSIDriverPolicy`.
8. Scegli Crea policy.

## Fase 2: creazione di un ruolo IAM

Il driver CSI di Mountpoint per Amazon S3 richiede le autorizzazioni Amazon S3 per interagire con il file system. In questa sezione viene descritto come creare un ruolo IAM per delegare queste autorizzazioni. Per creare questo ruolo, è possibile utilizzare uno di questi strumenti:

- the section called "eksctl"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

> **Note**
>
> La policy IAM `AmazonS3CSIDriverPolicy` è stata creata nella sezione precedente.

eksctl

creazione del ruolo IAM del driver CSI di Mountpoint per Amazon S3 con **`eksctl`**

Per creare il ruolo IAM e dell'account di servizio Kubernetes, esegui i comandi seguenti. Questi comandi allegano anche la policy IAM `AmazonS3CSIDriverPolicy` al ruolo, annotano l'account di servizio Kubernetes (`s3-csi-controller-sa`) con nome della risorsa Amazon (ARN) del ruolo IAM e aggiungono il nome dell'account di servizio Kubernetes alla policy di attendibilità per il ruolo IAM.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

Console di gestione AWS

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Select trusted entity (Seleziona entità attendibile), esegui le operazioni seguenti:
   a. Nella sezione Tipo di identità attendibile, scegli Identità Web.
   b. Per Identity provider (Provider di identità), scegli URL del provider OpenID Connect per il cluster (come mostrato nella scheda Overview (Panoramica) in Amazon EKS).

      Se non è visualizzato alcun URL, consulta Prerequisiti.
   c. Per Pubblico, scegli `sts.amazonaws.com`.
   d. Scegli Successivo.
5. Nella pagina Add permissions (Aggiungi autorizzazioni), esegui le operazioni seguenti:
   a. Nella casella Filtra policy, inserisci AmazonS3CSIDriverPolicy.

      > **Note**
      >
      > Questa policy è stata creata nella sezione precedente.

   b. Seleziona la casella di controllo a sinistra del risultato `AmazonS3CSIDriverPolicy` restituito dalla ricerca.
   c. Scegli Successivo.
6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   a. Per Nome del ruolo, inserisci un nome univoco per il tuo ruolo, ad esempio DriverRole Amazoneks_S3_CSI_.
   b. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   c. Scegli Crea ruolo.
7. Dopo aver creato il ruolo, sceglilo nella console in modo da aprirlo per la modifica.
8. Scegli la scheda Trust relationships (Relazioni di attendibilità), quindi scegli Edit trust policy (Modifica policy di attendibilità).

9. Cercare il risultato finale simile al seguente:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

   Inserisci una virgola alla fine della riga precedente, quindi aggiungine una seguente dopo quella riga. Sostituiscilo con la regione in cui si trova il cluster. *region-code* AWS Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* con l'ID del gestore OIDC del cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-driver-sa"
```

10. Assicurati che l'operatore `Condition` sia impostato su `"StringEquals"`.
11. Scegli Aggiorna policy per concludere.

AWS CLI

1. Visualizza l'URL del provider OIDC del cluster. Sostituisci *my-cluster* con il nome del cluster. Se l'output dal comando è `None`, rivedi i Prerequisiti.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

   Di seguito viene riportato un output di esempio.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Creare il ruolo IAM, concedendo all'account di servizio Kubernetes l'operazione `AssumeRoleWithWebIdentity`.
   a. Copiare i contenuti seguenti in un file denominato `aws-s3-csi-driver-trust-policy.json`. Sostituisci *111122223333* con l'ID del tuo account. Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* con i valori restituiti nella fase precedente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:s3-csi-
driver-sa",
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. Creare il ruolo. È possibile modificare *AmazonEKS_S3_CSI_DriverRole* con un nome diverso. In tal caso, assicurati di modificarlo anche nelle fasi successive.

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. Collega la policy IAM creata in precedenza al ruolo con il seguente comando.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> La policy IAM `AmazonS3CSIDriverPolicy` è stata creata nella sezione precedente.

4. Salta questo passaggio se stai installando il driver come componente aggiuntivo di Amazon EKS. Per le installazioni autogestite del driver, crea account di servizio Kubernetes annotati con l'ARN del ruolo IAM che hai creato.

   a. Salvare i seguenti contenuti in un file denominato `mountpoint-s3-service-account.yaml`. Sostituisci *111122223333* con l'ID del tuo account.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: s3-csi-driver-sa
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

b. Creare l'account del servizio Kubernetes nel cluster. L'account del servizio Kubernetes (`s3-csi-driver-sa`) è contrassegnato dal nome del ruolo IAM che hai creato. *`AmazonEKS_S3_CSI_DriverRole`*

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> Quando implementi il plugin tramite questa procedura, il plugin crea ed è configurato per l'utilizzo di un account di servizio denominato `s3-csi-driver-sa`.

## Fase 3: installazione del driver CSI di Mountpoint per Amazon S3

È possibile installare il driver CSI di Mountpoint per Amazon S3 tramite il componente aggiuntivo di Amazon EKS. Per aggiungere il componente aggiuntivo al cluster, è possibile utilizzare i seguenti strumenti:

- the section called "eksctl"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

In alternativa, è possibile installare il driver CSI Mountpoint per Amazon S3 come installazione autogestita. Per istruzioni su come eseguire un'installazione autogestita, consulta Installazione su GitHub.

A partire da `v1.8.0`, è possibile configurare i taint da tollerare per i pod del driver CSI. Per fare ciò, specifica un set personalizzato di contaminanti con `node.tolerations` cui tollerare o tollera tutti i contaminanti. `node.tolerateAllTaints` Per ulteriori informazioni, consulta Taint e tolleranze nella documentazione Kubernetes.

eksctl

Aggiungere il componente aggiuntivo CSI per Amazon S3 utilizzando **`eksctl`**

Eseguire il seguente comando seguente. Sostituisci *`my-cluster`* con il nome del cluster, *`111122223333`* con il tuo ID account e *`AmazonEKS_S3_CSI_DriverRole`* con il nome del ruolo IAM creato in precedenza.

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole --force
```

Se si rimuove l'opzione *`--force`* e tutte le impostazioni del componente aggiuntivo Amazon EKS sono in conflitto con le impostazioni esistenti, l'aggiornamento del componente aggiuntivo Amazon EKS non avrà successo. Verrà quindi visualizzato un messaggio di errore con la procedura utile a risolvere il conflitto. Prima di specificare questa opzione, assicurati che il componente aggiuntivo di Amazon EKS non gestisca le impostazioni che devi gestire tu, perché questa opzione le sovrascrive. Per ulteriori informazioni su altre opzioni per questa impostazione, consulta Componenti aggiuntivi nella documentazione di `eksctl`. Per ulteriori informazioni sulla gestione dei campi Kubernetes di Amazon EKS, consulta the section called "Campi che puoi personalizzare".

È possibile personalizzare `eksctl` tramite file di configurazione. Per ulteriori informazioni, consulta Funzionamento con i valori di configurazione nella documentazione `eksctl`. L'esempio seguente mostra come tollerare tutti i taint.

```
# config.yaml
...

addons:
- name: aws-mountpoint-s3-csi-driver
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
  configurationValues: |-
    node:
      tolerateAllTaints: true
```

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui configurare il componente aggiuntivo CSI di Mountpoint per Amazon S3.
4. Scegli la scheda Add-ons.
5. Scegli Ottieni altri componenti aggiuntivi.
6. Nella pagina Seleziona componenti aggiuntivi, procedi come segue:
   a. Nella EKS-addons  sezione  Amazon, seleziona la casella di controllo  Mountpoint for Amazon S3 CSI Driver.
   b. Scegli Avanti.
7. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, procedi come segue:
   a. Seleziona la Versione che desideri utilizzare.
   b. In Seleziona ruolo IAM, seleziona il nome di un ruolo IAM a cui è stata allegata la policy IAM del driver CSI di Mountpoint per Amazon S3.
   c. (Facoltativo) Aggiorna il Metodo di risoluzione dei conflitti dopo aver espanso le Impostazioni di configurazione opzionali. Se selezioni Sostituisci, una o più impostazioni per il componente aggiuntivo esistente possono essere sovrascritte con le impostazioni del componente aggiuntivo di Amazon EKS. Se non si abilita questa opzione e c'è un conflitto con le impostazioni esistenti, l'operazione non va a buon fine e viene visualizzato un messaggio di errore per aiutarti a risolvere il conflitto. Prima di selezionare questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni che devono essere gestite dall'utente.
   d. (Facoltativo) Configura le tolleranze nel campo Valori di configurazione dopo aver espanso le Impostazioni di configurazione opzionali.
   e. Scegli Avanti.
8. Nella pagina Rivedi e aggiungi, scegli Crea. Una volta completata l'installazione, viene visualizzato il componente aggiuntivo.

AWS CLI

Per aggiungere il componente aggiuntivo Mountpoint for Amazon S3 CSI utilizzando l'interfaccia a riga di comando AWS

Eseguire il seguente comando seguente. Sostituire *my-cluster* con il nome del cluster, *111122223333* con il proprio ID account e *AmazonEKS_S3_CSI_DriverRole* con il nome del ruolo creato in precedenza.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

È possibile personalizzare il comando con il contrassegno `--configuration-values`. L'esempio alternativo seguente mostra come tollerare tutti i taint.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole \
  --configuration-values '{"node":{"tolerateAllTaints":true}}'
```

### Fase 4: configurazione di Mountpoint per Amazon S3

Nella maggior parte dei casi, è possibile configurare Mountpoint per Amazon S3 solo con un nome di bucket. Per istruzioni sulla configurazione di Mountpoint per Amazon S3, consulta Configurazione di Mountpoint per Amazon S3 su. https://github.com/awslabs/mountpoint-s3/blob/main/doc/CONFIGURATION.md GitHub

### Fase 5: implementazione di un'applicazione esemplificativa

Puoi implementare il provisioning statico al driver su un bucket Amazon S3 esistente. Per ulteriori informazioni, consulta Static provisioning on. https://github.com/awslabs/mountpoint-s3-csi-driver/blob/main/examples/kubernetes/static_provisioning/README.md GitHub

## Rimozione del componente aggiuntivo Amazon EKS di Mountpoint per Amazon S3

Sono disponibili due opzioni per rimuovere il driver CSI di Mountpoint per Amazon S3.

- Mantenere il software aggiuntivo sul cluster – Questa opzione rimuove la gestione di Amazon EKS su qualsiasi impostazione. Inoltre, rimuove la possibilità per Amazon EKS di notificare gli aggiornamenti e aggiornare automaticamente il componente aggiuntivo Amazon EKS dopo l'avvio

di un aggiornamento. Tuttavia, mantiene il software aggiuntivo sul cluster. Questa opzione rende il componente aggiuntivo come autogestito anziché come un componente aggiuntivo Amazon EKS. Con questa opzione, non ci sono tempi di inattività per il componente aggiuntivo. I comandi in questa procedura utilizzano questa opzione.

- Rimuovere completamente il software aggiuntivo dal cluster – Consigliamo di rimuovere il componente aggiuntivo Amazon EKS dal cluster solo se non ci sono risorse nel cluster che dipendono dallo stesso. Per eseguire questa operazione, elimina `--preserve` dal comando utilizzato in questa procedura.

Se al componente aggiuntivo è associato un account IAM, l'account IAM non è rimosso.

È possibile utilizzare i seguenti strumenti per rimuovere il componente aggiuntivo CSI per Amazon S3:

- the section called "eksctl"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

### eksctl

Per rimuovere il componente aggiuntivo CSI per Amazon S3 utilizzando **`eksctl`**

Sostituire *`my-cluster`* con il nome del cluster ed eseguire il comando seguente.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --preserve
```

### Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui desideri rimuovere il componente aggiuntivo Mountpoint for Amazon S3 CSI.
4. Scegli la scheda Add-ons.
5. Scegli Driver CSI di Mountpoint per Amazon S3
6. Scegli Rimuovi.

7. Nella finestra di dialogo di conferma Rimuovi: aws-mountpoint-s3-csi-driver, procedi come segue:
   a. Se desideri che Amazon EKS interrompa la gestione delle impostazioni per il componente aggiuntivo, seleziona Mantieni sul cluster. Effettuare questa operazione se si desidera conservare il software aggiuntivo sul cluster. In questo modo sarà possibile gestire autonomamente tutte le impostazioni del componente aggiuntivo.
   b. Specificare `aws-mountpoint-s3-csi-driver`.
   c. Selezionare Rimuovi.

### AWS CLI

Per rimuovere il componente aggiuntivo Amazon S3 CSI utilizzando la CLI AWS

Sostituire *`my-cluster`* con il nome del cluster ed eseguire il comando seguente.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver --preserve
```

## Abilitare la funzionalità snapshot per i volumi CSI

La funzionalità snapshot consente di eseguire copie point-in-time dei dati. Affinché funzioni in Kubernetes, questa funzionalità richiede sia un driver CSI con supporto per snapshot (come il driver CSI di Amazon EBS), sia un controller di snapshot CSI. Il controller di snapshot è disponibile come componente aggiuntivo Amazon EKS o come installazione autogestita.

Di seguito sono elencati alcuni punti da considerare quando si utilizza il controller di snapshot CSI.

- Il controller di snapshot deve essere installato insieme a un driver CSI con funzionalità di snapshot. Per istruzioni sull'installazione del driver CSI di Amazon EBS, consultare la pagina the section called "Amazon EBS".
- Kubernetes non supporta snapshot di volumi serviti tramite migrazione CSI, come i volumi Amazon EBS che utilizzano una `StorageClass` con provisioner `kubernetes.io/aws-ebs`. I volumi devono essere creati con una `StorageClass` che fa riferimento al provisioner del driver CSI, `ebs.csi.aws.com`.
- La modalità automatica Amazon EKS non include il controller di snapshot. La funzionalità di archiviazione della modalità automatica EKS è compatibile con il controller di snapshot.

Consigliamo di installare il controller di snapshot CSI attraverso il componente aggiuntivo gestito di Amazon EKS. Questo componente aggiuntivo include le definizioni di risorse personalizzate (Custom Resource Definitions, CRD) necessarie per creare e gestire snapshot su Amazon EKS. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

In alternativa, se si desidera un'installazione autogestita del controller di snapshot CSI, consultare la pagina Usage nel `external-snapshotter` upstream Kubernetes su GitHub.

# Configurazione della rete per cluster Amazon EKS

Il cluster Amazon EKS viene creato in un VPC. Le reti pod sono fornite dal plug-in della CNI (Container Network Interface) di Amazon VPC per i nodi eseguiti sull'infrastruttura AWS. Se gestisci nodi sulla tua infrastruttura, consulta the section called "Configurazione della CNI". Per ulteriori informazioni sulle reti disponibili per il cluster, puoi consultare gli argomenti seguenti.

Argomenti


- Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione
- Visualizzazione di requisiti di rete di Amazon EKS per VPC e sottoreti
- Creazione di un Amazon VPC per il cluster Amazon EKS.
- Visualizzazione dei requisiti relativi al gruppo di sicurezza Amazon EKS per cluster
- Gestione di componenti aggiuntivi di rete per i cluster Amazon EKS


## Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione

1. Accedi al tuo cluster nella console di gestione.
2. Dalla scheda Rete, seleziona Gestisci risorse VPC.
3. Dal menu a discesa Sottoreti, seleziona sottoreti aggiuntive dal VPC del tuo cluster.

Per creare una nuova sottorete VPC:

- Consulta i requisiti per le sottoreti EKS.
- Consulta Create a Subnet nella Guida per l'utente di Amazon Virtual Private Cloud.

## Visualizzazione di requisiti di rete di Amazon EKS per VPC e sottoreti

Durante la creazione di un cluster, in genere si specificano un VPC e due sottoreti che si trovano in zone di disponibilità diverse. Questo argomento offre una panoramica dei requisiti e delle considerazioni specifiche di Amazon EKS per il VPC e le sottoreti utilizzate con il cluster. Se non si

dispone di un VPC da utilizzare con Amazon EKS, si veda the section called "Crea un VPC". Se stai creando un cluster locale o esteso su AWS Outposts, consulta the section called "Creazione di un VPC e delle sottoreti" invece questo argomento. Il contenuto di questo argomento si applica ai cluster Amazon EKS con nodi ibridi. Per ulteriori requisiti di rete per i nodi ibridi, consulta the section called "Preparazione della rete".

## Considerazioni e requisiti relativi al VPC

Durante la creazione di un cluster, il VPC specificato deve soddisfare i requisiti e le considerazioni seguenti:

- Il VPC deve disporre di un numero sufficiente di indirizzi IP da mettere a disposizione per il cluster, i nodi e le altre risorse Kubernetes da creare. In caso contrario, è possibile provare ad aumentare il numero di indirizzi IP disponibili.

  È possibile farlo aggiornando la configurazione del cluster per modificare le sottoreti e i gruppi di sicurezza utilizzati dal cluster. Puoi eseguire l' Console di gestione AWS aggiornamento dalla versione più recente della AWS CLI e dalla `eksctl` versione `v0.164.0-rc.0` o successiva. AWS CloudFormation Potrebbe essere necessario eseguire questa operazione per fornire alle sottoreti più indirizzi IP disponibili per aggiornare in modo corretto una versione del cluster.

  > **Important**
  >
  > Tutte le sottoreti aggiunte devono trovarsi nello stesso insieme di AZ fornito in origine quando hai creato il cluster. Le nuove sottoreti devono soddisfare tutti gli altri requisiti, ad esempio devono disporre di sufficienti indirizzi IP.
  > Supponiamo ad esempio che hai creato un cluster e specificato quattro sottoreti.
  > Nell'ordine in cui le hai specificate, la prima sottorete si trova nella zona di disponibilità `us-west-2a`, la seconda e la terza si trovano nella zona di disponibilità `us-west-2b` e la quarta si trova nella zona di disponibilità `us-west-2c`. Se desideri modificare le sottoreti, devi fornire almeno una sottorete in ciascuna delle tre zone di disponibilità e le sottoreti devono trovarsi nello stesso VPC delle sottoreti originali.

  Se hai bisogno di più indirizzi IP rispetto ai blocchi CIDR nel VPC, puoi aggiungere ulteriori blocchi CIDR associando ulteriori blocchi Classless Inter-Domain Routing (CIDR) al tuo VPC. L'associazione di blocchi di CIDR privati (RFC 1918) e pubblici (non RFC 1918) può avvenire prima o dopo la creazione del cluster.

È possibile aggiungere nodi che utilizzano il nuovo intervallo CIDR subito dopo averlo aggiunto. Tuttavia, poiché il piano di controllo riconosce il nuovo intervallo CIDR solo dopo il completamento della riconciliazione, il riconoscimento dell'intervallo CIDR associato a un VPC da parte del cluster può richiedere fino a un'ora. Quindi puoi eseguire i comandi`kubectl attach`, `kubectl cp` `kubectl exec`, `kubectl logs`, e (questi `kubectl port-forward` comandi usano il`kubelet API`) per nodi e pod nel nuovo blocco CIDR. Inoltre, se si dispone di pod che funzionano come backend webhook, devi attendere il completamento della riconciliazione del piano di controllo.

- Evita le sovrapposizioni degli intervalli di indirizzi IP quando connetti il tuo cluster EKS ad altri VPC tramite Transit Gateway, peering di VPC o altre configurazioni di rete. I conflitti di CIDR si verificano quando il servizio CIDR del cluster EKS si sovrappone al CIDR di un VPC connesso. In questi scenari, gli indirizzi IP del servizio hanno la priorità sulle risorse nei VPC connessi con lo stesso indirizzo IP, sebbene l'instradamento del traffico possa diventare imprevedibile e le applicazioni potrebbero non riuscire a connettersi alle risorse previste.

  Per evitare conflitti CIDR, assicurati che il tuo servizio CIDR di EKS non si sovrapponga a nessun CIDR di VPC connesso e mantieni un registro centralizzato di tutte le assegnazioni CIDR. Se si riscontrano sovrapposizioni di CIDR, è possibile utilizzare un gateway di transito con un VPC di servizi condivisi. Per ulteriori informazioni, consulta VPC isolati con servizi condivisi e Modelli di conservazione degli indirizzi IP instradabili di Amazon EKS VPC in una rete ibrida. Inoltre, consulta la sezione Comunicazione tra le sezioni di VPC della pagina Considerazioni su VPC e sottorete nella guida alle best practice di EKS.
- Se vuoi che Kubernetes assegni indirizzi `IPv6` a pod e servizi, associa un intervallo CIDR `IPv6` al tuo VPC. Per ulteriori informazioni, consulta Associazione di un blocco CIDR IPv6 al VPC nella Guida per l'utente di Amazon VPC. Non è possibile utilizzare indirizzi `IPv6` con pod e servizi in esecuzione su nodi ibridi e non è possibile utilizzare nodi ibridi con cluster configurati con la famiglia di indirizzi IP `IPv6`.
- Il VPC deve supportare il nome host `DNS` e la risoluzione `DNS`. In caso contrario, i nodi non possono registrarsi al cluster. Per ulteriori informazioni, consulta Attributi DNS nel VPC nella Guida per l'utente di Amazon VPC.
- Il VPC potrebbe richiedere l'utilizzo di endpoint VPC. AWS PrivateLink Per ulteriori informazioni, consulta the section called "Considerazioni e requisiti relativi alle sottoreti".
- Se il cluster utilizza il piano di controllo egress (`controlPlaneEgressMode=CUSTOMER_ROUTED`) indirizzato dal cliente, le sottoreti utilizzate dal cluster devono essere in grado di instradare verso qualsiasi endpoint chiamato dal piano di controllo, come i server webhook di ammissione e i provider OIDC. Ciò richiede un percorso di

uscita (ad esempio, un gateway NAT, un'istanza NAT, un firewall o un gateway di transito verso un VPC di uscita centralizzato) più una tabella di routing, un gruppo di sicurezza e regole ACL di rete che consentano il traffico. Per ulteriori informazioni, consulta the section called "Uscita del piano di controllo".

Se il cluster è stato creato con Kubernetes `1.14` o versioni precedenti, Amazon EKS aggiunge il tag seguente al VPC:

| Chiave | Valore |
| --- | --- |
| `kubernetes.io/cluster/` *`my-cluster`* | `owned` |

Questo tag è stato utilizzato solo da Amazon EKS, per cui è possibile rimuoverlo senza influire sui servizi. Non è necessario con la versione `1.15` del cluster o con versioni successive.

## Considerazioni e requisiti relativi alle sottoreti

Durante la creazione di un cluster, Amazon EKS crea da 2 a 4 interfacce di rete elastiche nelle sottoreti specificate. Queste interfacce di rete consentono la comunicazione tra il cluster e il VPC. Queste interfacce di rete abilitano anche le funzionalità di Kubernetes che utilizzano. `kubelet API` Le connessioni all'`kubelet`API vengono utilizzate nei comandi`kubectl attach`, `kubectl cp`kubectl `exec`, `kubectl logs` e. `kubectl port-forward` Ogni interfaccia di rete creata da Amazon EKS ha il testo `Amazon EKS` *`cluster-name`* nella sua descrizione.

Queste interfacce di rete gestite dal richiedente trasportano anche il traffico del piano dati tra il piano di controllo e i nodi di lavoro. Amazon EKS assorbe i costi di trasferimento dei dati sul lato del piano di controllo di queste interfacce di rete. Ti vengono addebitate le tariffe standard di trasferimento AWS dati per il traffico sul lato cliente, in particolare per quanto riguarda l'ingresso ai nodi di lavoro dal piano di controllo e l'uscita dai nodi di lavoro al piano di controllo. Per ulteriori informazioni, consulta the section called "Prezzi di Amazon EKS".

Amazon EKS può creare le proprie interfacce di rete in qualsiasi sottorete specificata al momento della creazione di un cluster. Puoi modificare le sottoreti in cui Amazon EKS crea le interfacce di rete dopo la creazione del cluster. Quando aggiorni la versione Kubernetes di un cluster, Amazon EKS elimina le interfacce di rete originali e ne crea di nuove Queste interfacce di rete potrebbero essere create all'interno delle stesse sottoreti o in sottoreti diverse rispetto alle interfacce di rete originali.

Per verificare in quali sottoreti vengono create le interfacce di rete, puoi limitare il numero di sottoreti specificate a due quando crei un cluster o aggiornare le sottoreti dopo aver creato il cluster.

## Requisiti relativi alla sottorete per i cluster

Le sottoreti specificate al momento della creazione o dell'aggiornamento di un cluster devono soddisfare i seguenti requisiti:

- Ciascuna sottorete deve disporre di almeno sei indirizzi IP per l'uso da parte di Amazon EKS. Tuttavia, consigliamo di utilizzare almeno 16 indirizzi IP.
- Le due sottoreti devono trovarsi in almeno due zone di disponibilità diverse.
- Le sottoreti non possono risiedere in Outposts o Wavelength. AWS AWS Tuttavia, se sono presenti nel VPC, puoi implementare nodi autogestiti e risorse Kubernetes per questi tipi di sottoreti. Per ulteriori informazioni sulla gestione dei nodi, consulta the section called "Nodi autogestiti".
- Le sottoreti possono essere pubbliche o private. Tuttavia, si consiglia di specificare sottoreti private, se possibile. Una sottorete pubblica è una sottorete con una tabella di routing che include un instradamento a un gateway Internet, mentre la sottorete privata non comprende tale instradamento.
- Le sottoreti non possono risiedere nelle seguenti zone di disponibilità:

| AWS Regione | Nome della Regione | ID della zona di disponibilità non consentita |
|---|---|---|
| `us-east-1` | Stati Uniti orientali (Virginia settentrionale) | `use1-az3` |
| `us-west-1` | Stati Uniti occidentali (California settentrionale) | `usw1-az2` |
| `ca-central-1` | Canada (Centrale) | `cac1-az3` |

## IP address family usage by component

La tabella seguente contiene la famiglia di indirizzi IP utilizzata da ciascun componente di Amazon EKS. È possibile utilizzare una network address translation (NAT) o un altro sistema di compatibilità per connettersi a questi componenti da indirizzi IP di origine in famiglie con il valore "No" per una voce di tabella.

La funzionalità può variare a seconda dell'impostazione della famiglia IP (`ipFamily`) del cluster. Questa impostazione modifica il tipo di indirizzi IP utilizzati per l'intervallo CIDR assegnato da Kubernetes ai servizi. Un cluster con il valore di impostazione IPv4 è definito come cluster IPv4 e un cluster con il valore di impostazione IPv6 è definito come cluster IPv6.

| Componente | Indirizzi IPv4 | Indirizzi IPv6 | Indirizzi dual stack |
|---|---|---|---|
| Endpoint pubblico dell'API EKS | Sì[1,3] | Sì[1,3] | Sì[1,3] |
| Endpoint VPC di API EKS | Sì | No | No |
| Endpoint pubblico di API di autentica zione EKS (EKS Pod Identity) | Sì[1] | Sì[1] | Sì[1] |
| Endpoint VPC di API di autenticazione EKS (EKS Pod Identity) | Sì[1] | Sì[1] | Sì[1] |
| `IPv4` Endpoint pubblico del cluster Kubernetes[2] | Sì | No | No |
| `IPv4` Endpoint privato del cluster Kubernete s[2] | Sì | No | No |
| `IPv6` Endpoint pubblico del cluster Kubernetes[2] | Sì[1,4] | Sì[1,4] | Sì[4] |
| `IPv6` Endpoint privato del cluster Kubernete s[2] | Sì[1,4] | Sì[1,4] | Sì[4] |

| Componente | Indirizzi IPv4 | Indirizzi IPv6 | Indirizzi dual stack |
|---|---|---|---|
| Sottoreti del cluster Kubernetes | Sì[2] | No | Sì[2] |
| Indirizzi IP primari del nodo | Sì[2] | No | Sì[2] |
| Intervallo CIDR del cluster per gli indirizzi IP del servizio | Sì[2] | Sì[2] | No |
| Indirizzi IP del pod da CNI di VPC | Sì[2] | Sì[2] | No |
| URL dell'emittente OIDC di IRSA | Sì[1,3] | Sì[1,3] | Sì[1,3] |

**Note**

[1] L'endpoint è dual-stack con gli indirizzi sia `IPv4` che `IPv6`. Le tue applicazioni esterne AWS, i tuoi nodi del cluster e i tuoi pod all'interno del cluster possono raggiungere questo endpoint in uno dei due `IPv4` modi. `IPv6`

[2] Si sceglie tra un cluster `IPv4` e un cluster `IPv6` nell'impostazione della famiglia IP (`ipFamily`) del cluster quando se ne crea uno e questa impostazione non può essere modificata. È invece necessario scegliere un'impostazione diversa quando si crea un altro cluster e si migrano i carichi di lavoro.

[3] L'endpoint dual-stack è stato introdotto nell'agosto 2024. Per utilizzare gli endpoint dual-stack con la AWS CLI, consulta la configurazione degli endpoint Dual-stack e FIPS nella Guida di riferimento per SDK e strumenti. AWS Di seguito sono elencati i nuovi endpoint:

endpoint pubblico dell'API EKS

```
eks.region.api.aws
```

URL dell'emittente OIDC di IRSA

```
oidc-eks.region.api.aws
```

[4] L'endpoint del cluster dual-stack è stato introdotto nell'ottobre 2024. EKS crea il seguente endpoint per i nuovi cluster creati dopo questa data e selezionati `IPv6` nell'impostazione della famiglia IP (IpFamily) del cluster:

Endpoint del cluster EKS public/private

```
eks-cluster.region.api.aws
```

## Requisiti relativi alla sottorete per i nodi

È possibile implementare nodi e risorse Kubernetes nelle stesse sottoreti specificate al momento della creazione del cluster. Tuttavia, non è necessario. di tali risorse può avvenire anche in sottoreti non specificate precedentemente. L'implementazione di nodi in sottoreti diverse non comporta la creazione di interfacce di rete cluster in tali sottoreti da parte di Amazon EKS. Qualsiasi sottorete in cui si implementano nodi e risorse Kubernetes deve soddisfare i requisiti seguenti:

- Le sottoreti devono mettere a disposizione un numero sufficiente di indirizzi IP per implementare tutti i nodi e le risorse Kubernetes.
- Se si desidera che Kubernetes assegni indirizzi `IPv6` a pod e servizi, è necessario disporre di un intervallo CIDR `IPv6` e un intervallo CIDR `IPv4` associati alla sottorete. Per ulteriori informazioni, consulta Associare un intervallo CIDR IPv6 alla sottorete nella guida per l'utente di Amazon VPC. Le tabelle di instradamento associate alle sottoreti devono includere instradamenti a indirizzi `IPv4` e `IPv6`. Per maggiori informazioni, consulta Routes (Instradamenti) nella Guida dell'utente di Amazon VPC. Ai pod viene assegnato solo un indirizzo `IPv6`, mentre alle interfacce di rete create da Amazon EKS per il cluster e ai nodi vengono assegnati un indirizzo `IPv4` e uno `IPv6`.
- Se è necessario garantire ai pod un accesso in entrata da Internet, assicurarsi di disporre di almeno una sottorete pubblica con un numero sufficiente di indirizzi IP disponibili su cui implementare i bilanciatori di carico e gli ingressi. È possibile implementare i load balancer nelle sottoreti pubbliche. I bilanciatori di carico possono bilanciare il carico sui pod di sottoreti private o pubbliche. Consigliamo di implementare i nodi su sottoreti private, se possibile.
- Se si prevede di implementare i nodi in una sottorete pubblica, quest'ultima deve essere configurata per assegnare automaticamente indirizzi `IPv4` pubblici o indirizzi `IPv6`. Se si implementano nodi in una sottorete privata a cui è associato un blocco CIDR `IPv6`, anche questa sottorete deve essere configurata per assegnare automaticamente indirizzi `IPv6`. Se hai utilizzato il AWS CloudFormation modello fornito da Amazon EKS per distribuire il tuo VPC dopo il 26 marzo

2020, questa impostazione è abilitata. Se hai utilizzato i modelli per implementare il VPC prima di tale data o utilizzi un VPC personale, devi abilitare questa impostazione manualmente. Per il modello, consulta the section called "Crea un VPC". Per ulteriori informazioni, consulta Modifica l'attributo di assegnazione degli indirizzi IPv4 pubblici per la sottorete e Modifica l'attributo di assegnazione degli indirizzi IPv6 per la sottorete nella guida per l'utente di Amazon VPC.

- Se la sottorete in cui distribuisci un nodo è una sottorete privata e la relativa tabella di routing non include una route verso un dispositivo NAT (Network Address Translation) () o un gateway di sola uscita (IPv4), aggiungi gli endpoint VPC utilizzando al tuo VPCIPv6. AWS PrivateLink I tuoi nodi e pod necessitano di endpoint VPC per tutti i servizi con cui comunicano. AWS Gli esempi includono Amazon ECR, Elastic Load Balancing, Amazon CloudWatch, AWS Security Token Service e Amazon Simple Storage Service (Amazon S3). Se utilizzi i ruoli IAM per gli account di servizio (IRSA), puoi anche raggiungere l'endpoint OIDC discovery/JWKS del cluster privatamente con l'endpoint. `com.amazonaws.region-code.oidc-eks` Per ulteriori informazioni, consulta the section called "Accedi all'endpoint OIDC del cluster utilizzando AWS PrivateLink". Non tutti i servizi supportano gli endpoint VPC. AWS Per ulteriori informazioni, consulta Che cos'è? AWS PrivateLink e AWS servizi che si integrano con AWS PrivateLink. Per un elenco di altri requisiti Amazon EKS, consulta the section called "Cluster privati".
- Se si desidera implementare i load balancer in una sottorete, quest'ultima deve presentare il tag seguente:
  - Sottoreti private

| Chiave | Valore |
| --- | --- |
| `kubernetes.io/role/internal-elb` | `1` |

  - Sottoreti pubbliche

| Chiave | Valore |
| --- | --- |
| `kubernetes.io/role/elb` | `1` |

In passato, quando si creava un cluster Kubernetes che è la versione `1.18` e precedenti, Amazon EKS aggiungeva il seguente tag a tutte le sottoreti specificate.

| Chiave | Valore |
| --- | --- |
| kubernetes.io/cluster/ *my-cluster* | shared |

Adesso, se crei un nuovo cluster Kubernetes, Amazon EKS non aggiunge alcun tag alle sottoreti. Se il tag era applicato a sottoreti utilizzate da un cluster che in precedenza era una versione precedente a `1.19`, il tag non veniva rimosso automaticamente dalle sottoreti quando il cluster veniva aggiornato a una versione più recente. La versione `2.1.1` o precedente del AWS Load Balancer Controller richiede questo tag. Se si utilizza una versione più recente di Load Balancer Controller, è possibile rimuovere il tag senza interrompere i servizi. Per ulteriori informazioni sul controller, consulta the section called " AWS Controller Load Balancer".

Se hai distribuito un VPC utilizzando `eksctl` uno qualsiasi dei modelli VPC di Amazon EKS, AWS CloudFormation vale quanto segue:

- In data 26 marzo 2020 o successiva. Gli indirizzi `IPv4` pubblici vengono assegnati automaticamente dalle sottoreti pubbliche ai nuovi nodi implementati nelle sottoreti pubbliche.
- Prima del 26 marzo 2020: gli indirizzi `IPv4` pubblici non vengono assegnati automaticamente dalle sottoreti pubbliche ai nuovi nodi implementati nelle sottoreti pubbliche.

Questa modifica influisce sui nuovi gruppi di nodi implementati nelle sottoreti pubbliche nei modi seguenti:

- Gruppi di nodi gestiti : se il gruppo di nodi viene implementato in una sottorete pubblica in data 22 aprile 2020 o successiva, la sottorete pubblica deve disporre dell'assegnazione automatica degli indirizzi IP pubblici abilitati. Per ulteriori informazioni, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete.
- Gruppi di nodi autogestiti Linux, Windows o Arm: se il gruppo di nodi viene implementato in una sottorete pubblica in data 26 marzo 2020 o successiva, la sottorete pubblica deve disporre dell'assegnazione automatica degli indirizzi IP pubblici abilitati. Altrimenti, i nodi devono essere avviati con un indirizzo IP pubblico. Per ulteriori informazioni, consultare Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete o Assegnare un indirizzo IPv4 pubblico durante l'avvio dell'istanza.

# Considerazioni e requisiti relativi alle sottoreti condivisi

Puoi utilizzare la condivisione  VPC  per condividere sottoreti con altri AWS account all'interno delle stesse organizzazioni. AWS È possibile creare cluster Amazon EKS in sottoreti condivise, tenendo a mente le seguenti considerazioni:

- Il proprietario della sottorete VPC deve condividere una sottorete con un account partecipante prima che tale account possa creare un cluster Amazon EKS al suo interno.
- Non è possibile avviare le risorse utilizzando il gruppo di sicurezza predefinito per il VPC, in quanto questo appartiene al proprietario. Inoltre, i partecipanti non possono avviare le risorse utilizzando i gruppi di sicurezza di proprietà di altri partecipanti o del proprietario.
- In una sottorete condivisa, il partecipante e il proprietario controllano separatamente i gruppi di sicurezza all'interno di ciascun rispettivo account. Il proprietario della sottorete può visualizzare i gruppi di sicurezza che sono stati creati dai partecipanti, ma non può eseguire operazioni sugli stessi. Se il proprietario della sottorete desidera rimuovere o modificare questi gruppi di sicurezza, è il partecipante che ha creato il gruppo di sicurezza che deve eseguire l'operazione.
- Se un cluster viene creato da un partecipante, valgono le seguenti considerazioni:
  - Il ruolo IAM del cluster e i ruoli IAM del nodo devono essere creati in quell'account. Per ulteriori informazioni, consulta the section called “Ruolo IAM del cluster” e the section called “Ruolo IAM del nodo”.
  - Tutti i nodi devono essere creati dallo stesso partecipante, inclusi i gruppi di nodi gestiti.
- Il proprietario del VPC condiviso non può visualizzare, aggiornare o eliminare un cluster creato da un partecipante nella sottorete condivisa. Ciò si aggiunge alle risorse VPC alle quali ogni account ha un accesso diverso. Per ulteriori informazioni, consulta la sezione Responsabilità e autorizzazioni per proprietari e partecipanti nella Guida per l'utente di Amazon VPC.
- Se si utilizza la funzionalità rete personalizzata del plugin CNI di Amazon VPC, è necessario utilizzare le mappature degli ID delle zone di disponibilità elencate nell'account del proprietario per creare ciascun `ENIConfig`. Per ulteriori informazioni, consulta the section called “Rete personalizzata”.

Per ulteriori informazioni sulla condivisione delle sottoreti VPC, consulta la pagina Condivisione del VPC con altri account nella Guida per l'utente di Amazon VPC.

# Creazione di un Amazon VPC per il cluster Amazon EKS.

Puoi usare Amazon Virtual Private Cloud (Amazon VPC) per lanciare AWS risorse in una rete virtuale che hai definito. Questa rete virtuale è simile a una rete tradizionale da gestire all'interno del proprio data center, ma con i vantaggi dell'infrastruttura scalabile di Amazon Web Services. Prima di implementare cluster Amazon EKS di produzione, è preferibile approfondire le nozioni relative al servizio Amazon VPC. Per ulteriori informazioni, consulta la Guida utente Amazon VPC.

Un cluster Amazon EKS, i nodi e le risorse Kubernetes vengono implementati su un VPC. Se desideri utilizzare un VPC esistente con Amazon EKS, tale VPC dovrà soddisfare i requisiti descritti nella sezione the section called "Requisiti del VPC e delle sottoreti". Questo argomento descrive come creare un VPC che soddisfi i requisiti di Amazon EKS utilizzando un modello fornito AWS CloudFormation da Amazon EKS. Dopo l'implementazione di un modello, puoi visualizzare le risorse create dal modello per sapere esattamente quali risorse ha creato e la configurazione di tali risorse. Se usi nodi ibridi, il tuo VPC deve avere dei percorsi nella tabella di routing per la tua rete on-premises. Per ulteriori informazioni sui requisiti di rete per i nodi ibridi, consulta the section called "Preparazione della rete".

## Prerequisiti

Per creare un VPC destinato ad Amazon EKS, devi disporre delle autorizzazioni IAM necessarie alla creazione delle risorse per Amazon VPC. Queste risorse includono VPC, sottoreti, gruppi di sicurezza, tabelle di instradamento, instradamenti e gateway sia Internet che NAT. Per ulteriori informazioni, consulta la sezione Creare un VPC con una sottorete pubblica nella Guida per l'utente di Amazon VPC e l'elenco completo di Azioni disponibile nella Guida di riferimento per l'autorizzazione del servizio.

È possibile creare un VPC con sottoreti sia pubbliche che private, solo pubbliche o solo private.

## Sottoreti pubbliche e private

Questo VPC dispone di due sottoreti pubbliche e due private. Alle sottoreti pubbliche è associata una tabella di routing con un routing a un gateway Internet. Tuttavia, la tabella di routing mento di una sottorete privata non ha una route a un gateway Internet. Sia la sottorete pubblica che la sottorete privata vengono distribuite nella stessa zona di disponibilità. Le altre sottoreti pubbliche e private vengono distribuite in una seconda zona di disponibilità nella stessa regione. AWS Si consiglia questa opzione per la maggior parte delle implementazioni,

dal momento che consente di distribuire i nodi in sottoreti private. Questa opzione consente a Kubernetes di implementare i bilanciatori del carico nelle sottoreti pubbliche, in grado di bilanciare il carico del traffico nei pod in esecuzione sui nodi nelle sottoreti private. Gli indirizzi `IPv4` pubblici vengono assegnati automaticamente ai nodi implementati nelle sottoreti pubbliche, tuttavia gli indirizzi `IPv4` pubblici non vengono assegnati ai nodi implementati nelle sottoreti private.

È anche possibile assegnare indirizzi `IPv6` anche ai nodi nelle sottoreti pubbliche e private. I nodi nelle sottoreti private possono comunicare con il cluster e altri servizi. AWS I pod possono comunicare con Internet tramite un gateway NAT utilizzando gli indirizzi `IPv4` o tramite un gateway Internet in uscita utilizzando gli indirizzi `IPv6` implementati in ciascuna zona di disponibilità. Viene implementato un gruppo di sicurezza che blocca il traffico in ingresso proveniente da origini diverse rispetto al cluster o ai nodi, consentendo al contempo il traffico in uscita da qualsiasi origine.
Le sottoreti vengono contrassegnate in modo che Kubernetes possa implementare i sistemi di bilanciamento del carico.

a. Apri la AWS CloudFormation console.
b. Dalla barra di navigazione, seleziona una AWS regione che supporta Amazon EKS.
c. Scegli Crea stack, Con nuove risorse (standard).
d. In Prerequisite - Prepare template (Prerequisito . Preparazione del modello), assicurarsi che Template is ready (Il modello è pronto) sia selezionato e quindi in Specify template (Specifica modello), selezionare Amazon S3 URL(URL Amazon S3).
e. Puoi creare un VPC che supporta solo `IPv4` o un VPC che supporta sia `IPv4` che `IPv6`. Incollare uno dei seguenti URL nell'area di testo Amazon S3 URL (URL Amazon S3) e scegliere Next (Successivo):
   - `IPv4`

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

- `IPv4` e `IPv6`

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

a. Nella pagina Specifica i dettagli dello stack immetti i relativi parametri, quindi scegli Next (Successivo).
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, puoi utilizzare il nome del modello utilizzato nella fase precedente. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
   - VpcBlock: scegli una gamma `IPv4` CIDR per il tuo VPC. A ogni nodo, pod e bilanciatore del carico che viene implementato viene assegnato un indirizzo `IPv4` da questo blocco. I valori `IPv4` di default forniscono indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificare questo comportamento. Per ulteriori informazioni, consultare VPC e dimensionamento della sottorete nella Guida per l'utente di Amazon VPC. È inoltre possibile aggiungere ulteriori blocchi CIDR al VPC una volta creato. Se si crea un VPC `IPv6`, gli intervalli CIDR `IPv6` vengono assegnati automaticamente dallo spazio Global Unicast Address di Amazon.
   - PublicSubnet01Block: Specificate un blocco `IPv4` CIDR per la sottorete pubblica 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.
   - PublicSubnet02Block: Specificare un blocco `IPv4` CIDR per la sottorete pubblica 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.
   - PrivateSubnet01Block: Specificare un blocco `IPv4` CIDR per la sottorete privata 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.
   - PrivateSubnet02Block: Specificare un blocco `IPv4` CIDR per la sottorete privata 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.
b. (Facoltativo) Nella pagina Configure stack options (Configura opzioni stack), aggiungere tag alle risorse dello stack, quindi scegliere Next (Successivo).
c. Nella pagina Revisione, scegliere Crea pila.
d. Quando viene creata la pila, selezionala nella console e scegli Outputs (Uscite).

e. Registra il VpcIdper il VPC che è stato creato. Questo valore sarà necessario al momento della creazione del cluster e dei nodi.

f. Registra il SubnetIdsper le sottoreti che sono state create e se le hai create come sottoreti pubbliche o private. Saranno necessari almeno due valori al momento della creazione del cluster e dei nodi.

g. Se è stato creato un VPC `IPv4`, ignorare questo passaggio. Se è stato creato un VPC `IPv6`, è necessario abilitare l'opzione di assegnazione automatica dell'indirizzo `IPv6` per le sottoreti pubbliche create dal modello. Tale impostazione è già abilitata per le sottoreti private. Per abilitare l'impostazione, completare la procedura seguente:

   i. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

   ii. Nel pannello di navigazione a sinistra, seleziona Subnets (Sottoreti).

   iii. Seleziona una delle sottoreti pubbliche (***stack-name***/SubnetPublic01 o***stack-name***/SubnetPublic02 contiene la parola public) e scegli Azioni, Modifica impostazioni di sottorete.

   iv. Selezionare la casella di controllo Enable auto-assign IPv6 address (Abilita l'assegnazione automatica dell'indirizzo IPv6) quindi scegliere Save (Salva).

   v. Esegui nuovamente i passaggi precedenti per l'altra sottorete pubblica.

## Solo sottoreti pubbliche

Questo VPC dispone di tre sottoreti pubbliche che vengono distribuite in diverse zone di disponibilità in una regione. AWS Tutti i nodi vengono assegnati automaticamente a indirizzi `IPv4` pubblici e possono inviare e ricevere traffico Internet tramite un gateway Internet. Viene implementato un gruppo di sicurezza che nega tutto il traffico in ingresso e consente tutto il traffico in uscita. Le sottoreti vengono contrassegnate in modo che Kubernetes possa implementare i sistemi di bilanciamento del carico.

a. Apri la AWS CloudFormation console.

b. Dalla barra di navigazione, seleziona una AWS regione che supporta Amazon EKS.

c. Scegliere Create stack (Crea pila), With new resources (standard) (Con nuove risorse (standard)).

d. In Preparazione del modello, assicurarsi che Template is ready (Il modello è pronto) sia selezionato e quindi in Origine del modello, selezionare Amazon S3 URL.

e. Incollare il seguente URL nell'area di testo URL Amazon S3 e scegliere Successivo:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

a. Nella pagina Specify Details (Specifica dettagli) immetti i relativi parametri, quindi scegli Next (Successivo).
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *amazon-eks-vpc-sample*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
   - VpcBlock: scegli un blocco CIDR per il tuo VPC. A ogni nodo, pod e bilanciatore del carico che viene implementato viene assegnato un indirizzo `IPv4` da questo blocco. I valori `IPv4` di default forniscono indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificare questo comportamento. Per ulteriori informazioni, consultare VPC e dimensionamento della sottorete nella Guida per l'utente di Amazon VPC. È inoltre possibile aggiungere ulteriori blocchi CIDR al VPC una volta creato.
   - Subnet01Block: specifica un blocco CIDR per la sottorete 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
   - Subnet02Block: specifica un blocco CIDR per sottorete 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
   - Subnet03Block: specifica un blocco CIDR per sottorete 3. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
b. (Facoltativo) Nella pagina Options (Opzioni), contrassegna con dei tag le risorse della pila. Scegli Next (Successivo).
c. Nella pagina Review (Revisione) scegli Create (Crea).
d. Quando viene creata la pila, selezionala nella console e scegli Outputs (Uscite).
e. Registra il VpcIdper il VPC che è stato creato. Questo valore sarà necessario al momento della creazione del cluster e dei nodi.
f. Registra il SubnetIdsper le sottoreti che sono state create. Saranno necessari almeno due valori al momento della creazione del cluster e dei nodi.

g. (Facoltativo) Qualsiasi cluster implementato su questo VPC è in grado di assegnare indirizzi `IPv4` privati ai tuoi pod e servizi. Se si desidera che i cluster implementati in questo VPC assegnino indirizzi `IPv6` privati a pod e servizi, è necessario apportare aggiornamenti al VPC, alla sottorete, alle tabelle di routing e ai gruppi di sicurezza. Per ulteriori informazioni, consultare Migrazione di VPC esistenti da IPv4 a IPv6 nella Guida per l'utente di Amazon VPC. Amazon EKS richiede che le sottoreti abbiano l'opzione degli indirizzi `Auto-assign IPv6` abilitata. Per impostazione predefinita, tale opzione è disattivata.

## Solo sottoreti private

Questo VPC dispone di tre sottoreti private che vengono distribuite in diverse zone di disponibilità della regione. AWS Le risorse implementate nelle sottoreti non possono accedere a Internet, né viceversa. Il modello crea endpoint VPC utilizzando diversi AWS servizi a cui AWS PrivateLink i nodi in genere devono accedere. Per consentire ai nodi di accedere a Internet in uscita, puoi aggiungere un gateway NAT pubblico nella zona di disponibilità di ogni sottorete dopo la creazione del VPC. Viene creato un gruppo di sicurezza che blocca tutto il traffico in entrata, ad eccezione delle risorse implementate nelle sottoreti. Un gruppo di sicurezza consente inoltre tutto il traffico in uscita. Le sottoreti vengono contrassegnate in modo che Kubernetes possa implementare i sistemi di bilanciamento del carico interno. Se stai creando un VPC con questa configurazione, consulta the section called "Cluster privati" per ulteriori requisiti e considerazioni.

a. Apri la AWS CloudFormation console.
b. Dalla barra di navigazione, seleziona una AWS regione che supporta Amazon EKS.
c. Scegliere Create stack (Crea pila), With new resources (standard) (Con nuove risorse (standard)).
d. In Preparazione del modello, assicurarsi che Template is ready (Il modello è pronto) sia selezionato e quindi in Origine del modello, selezionare Amazon S3 URL.
e. Incollare il seguente URL nell'area di testo URL Amazon S3 e scegliere Successivo:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

a. Nella pagina Specify Details (Specifica dettagli), immetti i relativi parametri, quindi scegli Next (Successivo).
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *`amazon-eks-fully-private-vpc`*. Il nome può contenere solo caratteri

alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.

- VpcBlock: scegli un blocco CIDR per il tuo VPC. A ogni nodo, pod e bilanciatore del carico che viene implementato viene assegnato un indirizzo `IPv4` da questo blocco. I valori `IPv4` di default forniscono indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificare questo comportamento. Per ulteriori informazioni, consultare VPC e dimensionamento della sottorete nella Guida per l'utente di Amazon VPC. È inoltre possibile aggiungere ulteriori blocchi CIDR al VPC una volta creato.
- PrivateSubnet01Block: Specificate un blocco CIDR per la sottorete 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
- PrivateSubnet02Block: Specificare un blocco CIDR per la sottorete 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
- PrivateSubnet03Block: Specificare un blocco CIDR per la sottorete 3. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.

b. (Facoltativo) Nella pagina Options (Opzioni), contrassegna con dei tag le risorse della pila. Scegli Next (Successivo).

c. Nella pagina Review (Revisione) scegli Create (Crea).

d. Quando viene creata la pila, selezionala nella console e scegli Outputs (Uscite).

e. Registra il VpcIdper il VPC che è stato creato. Questo valore sarà necessario al momento della creazione del cluster e dei nodi.

f. Registra il SubnetIdsper le sottoreti che sono state create. Saranno necessari almeno due valori al momento della creazione del cluster e dei nodi.

g. (Facoltativo) Qualsiasi cluster implementato su questo VPC è in grado di assegnare indirizzi `IPv4` privati ai tuoi pod e servizi. Se si desidera che i cluster implementati in questo VPC assegnino indirizzi `IPv6` privati a pod e servizi, è necessario apportare aggiornamenti al VPC, alla sottorete, alle tabelle di routing e ai gruppi di sicurezza. Per ulteriori informazioni, consultare Migrazione di VPC esistenti da IPv4 a IPv6 nella Guida per l'utente di Amazon VPC. Amazon EKS richiede che le sottoreti abbiano l'opzione degli indirizzi `Auto-assign IPv6`abilitata (disattivata per impostazione predefinita).

# Visualizzazione dei requisiti relativi al gruppo di sicurezza Amazon EKS per cluster

In questo argomento vengono descritti i requisiti relativi al gruppo di sicurezza per un cluster Amazon EKS.

## Gruppo di sicurezza del cluster predefinito

Quando si crea un cluster, Amazon EKS crea un gruppo di sicurezza denominato `eks-cluster-sg-`*`my-cluster`*`-`*`uniqueID`* . Questo gruppo di sicurezza presenta le regole predefinite seguenti:

| Tipo di regola | Protocollo | Porte | Origine | Destinazione |
|---|---|---|---|---|
| In entrata | Tutti | Tutti | Personale | |
| In uscita | Tutti | Tutti | | 0.0.0.0/0(IPv4) o ::/0 (IPv6) |
| In uscita | Tutti | Tutti | | Self (per il traffico EFA) |

Il gruppo di sicurezza predefinito include una regola in uscita che consente il traffico di Elastic Fabric Adapter (EFA) con la destinazione dello stesso gruppo di sicurezza. Ciò consente il traffico EFA all'interno del cluster, il che è vantaggioso per i AI/ML carichi di lavoro HPC (High Performance Computing). Per ulteriori informazioni, consulta Elastic Fabric Adapter per AI/ML carichi di lavoro HPC su Amazon EC2 nella Amazon Elastic Compute Cloud User Guide.

> **Important**
>
> Se il tuo cluster non necessita della regola in uscita per `0.0.0.0/0` (IPv4), `::/0` (IPv6), puoi rimuoverla. Se la rimuovi, dovrai comunque avere le regole minime elencate in Limitazione del traffico del cluster. Se rimuovi le regole in entrata o in uscita che consentono to/from il traffico verso il gruppo di sicurezza del cluster stesso, Amazon EKS le ricrea ogni volta che il cluster viene aggiornato.

Amazon EKS aggiunge i seguenti tag al gruppo di sicurezza. Se rimuovi i tag, Amazon EKS li aggiunge nuovamente al gruppo di sicurezza ogni volta che il cluster viene aggiornato.

| Chiave | Valore |
| --- | --- |
| kubernetes.io/cluster/ *my-cluster* | owned |
| aws:eks:cluster-name | *my-cluster* |
| Name | eks-cluster-sg- *my-cluster* -*uniqueid* |

Amazon EKS crea le risorse riportate di seguito e le associa automaticamente a questo gruppo di sicurezza:

- 2-4 interfacce di rete elastiche (indicate nel resto del documento come interfacce di rete) create insieme al cluster.
- Interfacce di rete dei nodi in qualsiasi gruppo di nodi gestito creato.

Le regole predefinite consentono il flusso del traffico tra il cluster e i nodi e il traffico in uscita verso qualsiasi destinazione. Quando crei un cluster, puoi specificare i gruppi di sicurezza personali se lo desideri. In tal caso, Amazon EKS associa i gruppi di sicurezza specificati alle interfacce di rete create per il cluster, Tuttavia, non li associa ai gruppi di nodi che crei.

Puoi determinare l'ID del gruppo di sicurezza del cluster nella Console di gestione AWS sezione Rete del cluster. In alternativa, puoi farlo eseguendo il seguente comando AWS CLI.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

## Limitazione del traffico cluster

Se è necessario limitare le porte aperte tra il piano di controllo EKS e i nodi, è possibile rimuovere la regola di uscita predefinita a `0.0.0.0/0` (IPv4)/`::/0`(IPv6) e aggiungere le seguenti regole minime necessarie per il cluster.

Se rimuovi la regola in entrata predefinita che consente tutto il traffico per l'origine autonoma (traffico proveniente dal gruppo di sicurezza del cluster), Amazon EKS la ricrea quando il cluster viene aggiornato.

Se rimuovi la regola in uscita predefinita che consente tutto il traffico per la destinazione automatica (traffico verso il gruppo di sicurezza del cluster), Amazon EKS la ricrea quando il cluster viene aggiornato.

| Tipo di regola | Protocollo | Porta | Destinazione |
|---|---|---|---|
| In uscita | TCP | 443 | Gruppo di sicurezza del cluster |
| In uscita | TCP | 10250 | Gruppo di sicurezza del cluster |
| In uscita (DNS) | TCP e UDP | 53 | Gruppo di sicurezza del cluster |

È inoltre necessario aggiungere regole per il traffico seguente:

- Qualsiasi protocollo e porta che si prevede di utilizzare per la comunicazione tra i nodi.
- Accesso a Internet in uscita in modo che i nodi possano accedere ad Amazon EKS APIs per l'introspezione dei cluster e la registrazione dei nodi al momento del lancio. Se i nodi non hanno accesso a Internet, consulta Implementazione di cluster privati con accesso Internet limitato per ulteriori considerazioni.
- Accesso al nodo per estrarre immagini di container da Amazon ECR o da altri registri di container da APIs cui devono estrarre immagini, ad esempio. DockerHub Per ulteriori informazioni, consulta Intervalli di indirizzi AWS IP nella AWS Guida generale.
- Accesso dei nodi ad Amazon S3.
- Sono necessarie regole separate per gli indirizzi `IPv4` e `IPv6`.
- Se utilizzi nodi ibridi, devi aggiungere un gruppo di sicurezza aggiuntivo al cluster per consentire la comunicazione con i nodi on-premises e i pod. Per ulteriori informazioni, consulta the section called "Preparazione della rete".

Se stai pianificando una limitazione delle regole, ti consigliamo di testare accuratamente tutti i pod prima di applicare le regole modificate a un cluster di produzione.

Se precedentemente hai implementato un cluster con Kubernetes `1.14` e una versione della piattaforma `eks.3` o precedente, considera quanto segue:

- È probabile che siano presenti anche piani di controllo (control-plane) e gruppi di sicurezza dei nodi. Al momento della creazione, questi gruppi contenevano le regole con restrizioni elencate nella tabella precedente, che adesso possono essere rimosse in quanto non più necessarie. Tuttavia, è necessario assicurarsi che il gruppo di sicurezza del cluster contenga le regole incluse in tali gruppi.
- Se hai distribuito il cluster utilizzando direttamente l'API o hai utilizzato uno strumento come la AWS CLI AWS CloudFormation o per creare il cluster e non hai specificato un gruppo di sicurezza al momento della creazione del cluster, il gruppo di sicurezza predefinito per il VPC è stato applicato alle interfacce di rete del cluster create da Amazon EKS.

## Gruppi di sicurezza condivisi

Amazon EKS supporta gruppi di sicurezza condivisi.

- Le associazioni VPC dei gruppi di sicurezza associano i gruppi di sicurezza VPCs a più gruppi nello stesso account e nella stessa regione.
  - Scopri come associare gruppi di sicurezza a più gruppi di sicurezza VPCs nella Amazon VPC User Guide.
- I gruppi di sicurezza condivisi consentono di condividere gruppi di sicurezza con altri AWS account. Gli account devono appartenere alla stessa AWS organizzazione.
  - Scopri come condividere i gruppi di sicurezza con le organizzazioni nella Guida per l'utente di Amazon VPC.
- I gruppi di sicurezza sono sempre limitati a una singola AWS regione.

### Considerazioni per Amazon EKS

- EKS ha gli stessi requisiti dei gruppi di sicurezza condivisi o multi-VPC dei gruppi di sicurezza standard.

# Gestione di componenti aggiuntivi di rete per i cluster Amazon EKS

Per il cluster Amazon EKS sono disponibili diversi componenti aggiuntivi di rete.

## Built-in componenti aggiuntivi

> **Note**
>
> Quando crei un cluster EKS:
>
> - Utilizzo della AWS console: i componenti aggiuntivi integrati (come CoredNS, kube-proxy, ecc.) vengono installati automaticamente come Amazon EKS. Add-ons Questi possono essere facilmente configurati e aggiornati tramite la AWS console, la CLI o gli SDK.
> - Utilizzando altri metodi (CLI, SDK, ecc.): gli stessi componenti aggiuntivi integrati vengono installati come versioni autogestite che vengono eseguite come normali implementazioni di Kubernetes. Questi richiedono configurazioni e aggiornamenti manuali poiché non possono essere gestiti tramite AWS strumenti.
>
> Consigliamo di utilizzare Amazon EKS Add-ons anziché le versioni autogestite per semplificare la gestione dei componenti aggiuntivi e consentire la configurazione e gli aggiornamenti centralizzati tramite i servizi. AWS

Plug-in della CNI di Amazon VPC per Kubernetes

Questo componente aggiuntivo CNI crea interfacce di rete elastiche e le collega ai nodi Amazon EC2. Il componente aggiuntivo, inoltre, assegna un indirizzo `IPv4` o `IPv6` privato dal VPC a ogni pod e servizio. Per impostazione predefinita, nel cluster è stato installato questo componente aggiuntivo. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC". Se utilizzi nodi ibridi, la CNI del VPC è ancora installata per impostazione predefinita, ma non può essere eseguita sui nodi ibridi con una regola di anti-affinità. Per ulteriori informazioni sulle opzioni della CNI per i nodi ibridi, consulta the section called "Configurazione della CNI".

CoreDNS

CoreDNS è un server DNS flessibile ed estensibile che può fungere da cluster DNS Kubernetes. CoreDNS fornisce la risoluzione dei nomi per tutti i pod del cluster. Per impostazione predefinita, nel cluster è stato installato questo componente aggiuntivo. Per ulteriori informazioni, consulta the section called "CoreDNS".

**`kube-proxy`**

Questo componente aggiuntivo mantiene le regole di rete sui nodi Amazon EC2 e consente la comunicazione di rete con i tuoi pod. Per impostazione predefinita, nel cluster è stato installato questo componente aggiuntivo. Per ulteriori informazioni, consulta the section called "kube-proxy".

## Facoltativo AWS componenti aggiuntivi di rete

AWS Load Balancer Controller

Quando si distribuiscono oggetti di servizio Kubernetes di tipo diverso`loadbalancer`, il controller crea Network Load Balancer. AWS Quando crei oggetti di ingresso Kubernetes, il controller crea Application Load Balancer. AWS Consigliamo di utilizzare questo controller per il provisioning di Network Load Balancer, anziché utilizzare il controller Cloud Provider legacy integrato in Kubernetes. Per ulteriori informazioni, consultare la documentazione sul Controller del load balancer AWS.

AWS Gateway API Controller

Questo controller consente di connettere i servizi su più cluster Kubernetes utilizzando l'API gateway Kubernetes. Il controller collega i servizi Kubernetes in esecuzione su istanze, container e funzioni serverless Amazon EC2, utilizzando il servizio Amazon VPC Lattice. Per ulteriori informazioni, consulta documentazione Controller API Gateway AWS.

Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

## Assegna IP ai pod con CNI di Amazon VPC

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS.

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non devi installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.

Per ulteriori informazioni, consulta Modalità automatica EKS.

Il plug-in CNI di Amazon VPC per il componente aggiuntivo Kubernetes viene implementato su ogni nodo Amazon EC2 del cluster Amazon EKS. Il componente aggiuntivo crea interfacce di rete elastiche e le collega ai nodi Amazon EC2. Il componente aggiuntivo, inoltre, assegna un indirizzo `IPv4` o `IPv6` privato dal VPC a ogni pod.

Una versione del componente aggiuntivo viene implementata con ogni nodo Fargate nel cluster, ma non viene aggiornata sui nodi Fargate. Altri plug-in CNI compatibili sono disponibili per l'uso sui cluster Amazon EKS, ma questo è l'unico plug-in CNI supportato da Amazon EKS per i nodi che funzionano sull'infrastruttura. AWS Per ulteriori informazioni sugli altri plug-in CNI compatibili, consulta the section called "Plug-in CNI alternativi". Il CNI di VPC non è supportato per l'utilizzo con nodi ibridi. Per ulteriori informazioni sulle opzioni della CNI per i nodi ibridi, consulta the section called "Configurazione della CNI".

La tabella seguente elenca la versione più recente del componente aggiuntivo di Amazon EKS disponibile per ogni versione di Kubernetes.

## Versioni del CNI di Amazon VPC

| Versione di Kubernetes | Tipo di VPC di Amazon EKS versione CNI |
|---|---|
| 1.36 | v1.22.4-eksbuild.3 |
| 1,35 | v1.22.4-eksbuild.3 |
| 1,34 | v1.22.4-eksbuild.3 |
| 1.33 | v1.22.4-eksbuild.3 |
| 1,32 | v1.22.4-eksbuild.3 |
| 1,31 | v1.22.4-eksbuild.3 |

**Important**

Se gestisci autonomamente questo componente aggiuntivo, le versioni nella tabella potrebbero non essere le stesse di quelle autogestite disponibili. Per ulteriori informazioni

sull'aggiornamento del componente aggiuntivo del tipo autogestito, consulta the section called "Aggiornamento (autogestito)".

> **Important**
>
> Per eseguire l'aggiornamento al CNI VPC versione v1.12.0 o successiva, devi prima eseguire l'aggiornamento alla versione v1.7.0. È consigliabile aggiornare solo una versione secondaria alla volta.

## Considerazioni

Di seguito sono riportate alcune considerazioni sull'utilizzo della funzionalità.

- Le versioni sono specificate come `major-version.minor-version.patch-version-eksbuild.build-number`.
- Verifica della compatibilità delle versioni per le varie funzionalità. Alcune funzionalità di ogni versione del plug-in CNI di Amazon VPC per Kubernetes richiedono versioni specifiche di Kubernetes. Quando si utilizzano diverse funzionalità di Amazon EKS ed è necessaria una versione specifica del componente aggiuntivo, questa viene annotata nella documentazione relativa alle funzionalità. A meno che tu non abbia un motivo specifico per eseguire una versione più vecchia, consigliamo di eseguire la versione più recente.

## Creazione di CNI di Amazon VPC (componente aggiuntivo di Amazon EKS)

Utilizza i seguenti passaggi per creare il plugin CNI di Amazon VPC per il componente aggiuntivo Kubernetes di Amazon EKS.

Prima di iniziare, esamina le seguenti informazioni. Per ulteriori informazioni, consulta the section called "Considerazioni".

Prerequisiti

Di seguito sono riportati i seguenti prerequisiti per il plugin CNI di Amazon VPC per Kubernetes di Amazon EKS.

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.

- Un provider di AWS Identity and Access Management (IAM) OpenID Connect (OIDC) esistente per il tuo cluster. Per determinare se si dispone già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Un ruolo IAM con la policy IAM Amazoneks_CNI_Policy (se il cluster utilizza la famiglia) o una policy (se il cluster utilizza `IPv4` la famiglia) ad esso allegata. IPv6 `IPv6` Per ulteriori informazioni sul ruolo CNI di VPC, consulta the section called "Configurare per IRSA". Per informazioni sulla politica, consulta. IPv6 the section called "`Crea una policy IAM per i cluster che utilizzano la famiglia IPv6`"

> **Important**
>
> Plugin CNI di Amazon VPC per versioni Kubernetes `v1.16.0` per implementare `v1.16.1` versioni `v1.0.0` con specifiche di CNI. Per ulteriori informazioni sulle `v1.0.0` specifiche CNI, vedere la specifica CNI (Container Network Interface) su. GitHub

Procedura

Una volta completati i seguenti prerequisiti, segui i passaggi di seguito per creare il componente aggiuntivo.

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster.

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.16.4-eksbuild.2
   ```

2. Scopri qual è il tipo di componente aggiuntivo attualmente installato sul cluster. A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Sostituisci *`my-cluster`* con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

Se è restituito il numero di versione, sul cluster è installato il componente aggiuntivo del tipo Amazon EKS e non è necessario completare i passaggi rimanenti di questa procedura. Se è restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completa i passaggi rimanenti di questa procedura per installarlo.

3. Salva la configurazione del componente aggiuntivo attualmente installato.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Crea il componente aggiuntivo utilizzando la AWS CLI. Se desideri utilizzare Console di gestione AWS o `eksctl` creare il componente aggiuntivo, consulta the section called "Creare un componente aggiuntivo" e specifica il nome del componente `vpc-cni` aggiuntivo. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.
   - Sostituisci *`my-cluster`* con il nome del cluster.
   - *`v1.20.3-eksbuild.1`*Sostituiscilo con la versione più recente elencata nella tabella delle versioni più recenti per la versione del cluster. Per la tabella con la versione più recente, consulta the section called "Versioni del CNI di Amazon VPC".
   - Sostituiscilo *`111122223333`* con l'ID del tuo account e *`AmazonEKSVPCCNIRole`* con il nome di un ruolo IAM esistente che hai creato. Per specificare un ruolo è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.3-eksbuild.1 \
    --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
```

   Se hai applicato impostazioni personalizzate al tuo attuale componente aggiuntivo che sono in conflitto con le impostazioni predefinite del componente aggiuntivo di Amazon EKS, la creazione potrebbe non riuscire. Se la creazione non riesce, riceverai un errore che può aiutare a risolvere il problema. In alternativa, puoi aggiungere `--resolve-conflicts OVERWRITE` al comando precedente. Ciò consente al componente aggiuntivo di sovrascrivere le impostazioni personalizzate esistenti. Una volta creato il componente aggiuntivo, è possibile aggiornarlo con le tue impostazioni personalizzate.

5. Verifica che la versione più recente del componente aggiuntivo per la versione del cluster sia stata aggiunta al cluster. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

La creazione del componente aggiuntivo potrebbe richiedere alcuni secondi.

Di seguito viene riportato un output di esempio.

```
v1.20.3-eksbuild.1
```

6. Se hai creato impostazioni personalizzate per il componente aggiuntivo originale, prima di creare il componente aggiuntivo di Amazon EKS, utilizza la configurazione che hai salvato precedentemente per aggiornare il componente aggiuntivo di Amazon EKS con le tue impostazioni personalizzate. Segui la procedura riportata in the section called "Aggiornamento (componente aggiuntivo EKS)".
7. (Facoltativo) Installa `cni-metrics-helper` sul tuo cluster. Raccoglie le informazioni sull'interfaccia di rete elastica e sull'indirizzo IP, le aggrega a livello di cluster e pubblica le metriche su Amazon. CloudWatch Per ulteriori informazioni, consulta on. cni-metrics-helper GitHub

## Aggiornamento di CNI di Amazon VPC (componente aggiuntivo Amazon EKS)

Aggiorna il tipo Amazon EKS del componente aggiuntivo del plugin CNI di Amazon VPC per Kubernetes del componente aggiuntivo Amazon EKS. Se non hai aggiunto il tipo di componente aggiuntivo Amazon EKS al cluster, è possibile installarlo seguendo the section called "Crea". In alternativa, aggiorna l'altro tipo di installazione CNI di VPC seguendo the section called "Aggiornamento (autogestito)".

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

Di seguito viene riportato un output di esempio.

```
v1.20.0-eksbuild.1
```

Confronta la versione con la tabella delle versioni più recenti all'indirizzo the section called "Versioni del CNI di Amazon VPC". Se la versione restituita è la stessa della versione Kubernetes del cluster presente nella tabella delle versioni più recenti, la versione più recente è già installata nel cluster e non è necessario completare il resto di questa procedura. Se nell'output ricevi un errore invece di un numero di versione, il componente aggiuntivo del tipo Amazon EKS non è installato sul cluster. È necessario creare il componente aggiuntivo prima di poterlo aggiornare con questa procedura. Per creare il tipo Amazon EKS del componente aggiuntivo CNI di VPC, segui questa pagina the section called "Crea".

2. Salva la configurazione del componente aggiuntivo attualmente installato.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. Aggiorna il tuo componente aggiuntivo utilizzando la AWS CLI. Se desideri utilizzare Console di gestione AWS o `eksctl` aggiornare il componente aggiuntivo, consulta. the section called "Aggiornamento di un componente aggiuntivo" Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.
   - Sostituisci *`my-cluster`* con il nome del cluster.
   - Sostituiscilo *`v1.20.0-eksbuild.1`* con la versione più recente elencata nella tabella delle versioni più recenti della versione del cluster.
   - Sostituiscilo *`111122223333`* con l'ID del tuo account e *`AmazonEKSVPCCNIRole`* con il nome di un ruolo IAM esistente che hai creato. Per creare un ruolo IAM per CNI di VPC, consulta the section called "Passaggio 1: Creazione di un ruolo IAM per il plug-in CNI di Amazon VPC per Kubernetes". Per specificare un ruolo è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
   - L'opzione `--resolve-conflicts PRESERVE` mantiene i valori di configurazione esistenti per il componente aggiuntivo. Se sono stati configurati valori personalizzati per le impostazioni del componente aggiuntivo e non si utilizza questa opzione, Amazon EKS sovrascrive i valori personalizzati con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `OVERWRITE`, tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in `none`, Amazon EKS non modifica il

valore di alcuna impostazione, ma l'aggiornamento potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.

- Se non viene aggiornata un'impostazione di configurazione, rimuovere `--configuration-values '{`*`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`*`}'` dal comando. Se stai aggiornando un'impostazione di configurazione, sostituiscila *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* con l'impostazione che desideri impostare. In questo esempio, la variabile di ambiente `AWS_VPC_K8S_CNI_EXTERNALSNAT` è impostata su `true`. Il valore specificato deve essere valido per lo schema di configurazione. Se non conosci lo schema di configurazione`aws eks describe-addon-configuration --addon-name vpc-cni --addon-version` *`v1.20.0-eksbuild.1`* , eseguilo sostituendolo *`v1.20.0-eksbuild.1`* con il numero di versione del componente aggiuntivo di cui vuoi vedere la configurazione. Lo schema viene restituito nell'output. Se disponi di una configurazione personalizzata esistente, desideri rimuoverla tutta e ripristinare i valori per tutte le impostazioni ai valori predefiniti di Amazon EKS, rimuovila *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* dal comando, in modo che sia vuota. `{}` Per una spiegazione di ciascuna impostazione, consulta Variabili di configurazione CNI su. GitHub

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.3-eksbuild.1 \
    --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

4. Conferma che la versione del componente aggiuntivo sia stata aggiornata. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

Di seguito viene riportato un output di esempio.

```
{
    "addon": {
        "addonName": "vpc-cni",
```

```
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.20.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws: eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn": "arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

Risoluzione dei problemi

Quando si aggiorna VPC CNI da una versione precedente alla v1.13.2, è necessario sostituire tutti i nodi del cluster dopo l'aggiornamento. Le versioni precedenti alla v1.13.2 utilizzano il backend iptables-legacy per inserire le regole iptables necessarie per il corretto funzionamento, come il codice sorgente NAT (SNAT).

La versione v1.13.2 è stata una release importante che ha introdotto iptables-wrapper, che rileva automaticamente il backend iptables appropriato (iptables-legacy o iptables-nft) per l'inserimento di catene e regole. Questa modifica è in linea con la decisione iniziale di Kubernetes di abbandonare il backend legacy a causa delle limitazioni delle prestazioni.

La sostituzione dei nodi dopo un aggiornamento da una versione precedente alla v1.13.2 di VPC CNI è necessaria perché l'introduzione di regole nei backend iptables-legacy e iptables-nft può portare a comportamenti imprevisti per il traffico proveniente da fonti non primarie. ENIs

## Aggiornamento di CNI di Amazon VPC (componente aggiuntivo autogestito)

> **Important**
>
> Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. Se non sono ben chiare le differenze tra i due tipi, consulta the

section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non riesci a utilizzare il componente aggiuntivo Amazon EKS, ti consigliamo di segnalare un problema sul motivo per cui non puoi farlo all'archivio della roadmap GitHub di Containers.

1. Verifica che sul cluster non sia installato il tipo di Amazon EKS del componente aggiuntivo. Sostituisci *my-cluster* con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

   Se è restituito un messaggio di errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Per gestire automaticamente il componente aggiuntivo, completa i passaggi rimanenti di questa procedura per aggiornare il componente aggiuntivo. Se viene restituito il numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo. Per aggiornarlo, utilizza la procedura descritta in the section called "Aggiornamento di un componente aggiuntivo" anziché questa. Se non conosci bene le differenze tra i due tipi di componente aggiuntivo, consulta the section called "Componenti aggiuntivi Amazon EKS".
2. Scopri qual è la versione dell'immagine di container attualmente installata sul cluster.

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.20.0-eksbuild.1
   ```

   L'output potrebbe non includere il numero di build.
3. Effettua il backup delle impostazioni correnti in modo da poter riprodurre la medesima configurazione dopo avere aggiornato la versione.

   ```
   kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
   ```

   Per esaminare le versioni disponibili e acquisire familiarità con le modifiche nella versione a cui desideri eseguire l'aggiornamento, consulta le versioni su. GitHub Tieni presente che ti

consigliamo di eseguire l'aggiornamento alla stessa. `major` `minor`. `patch`versione elencata nella tabella delle ultime versioni disponibili, anche se le versioni successive sono disponibili su GitHub. Per la tabella delle versioni più recenti disponibili, consulta the section called "Versioni del CNI di Amazon VPC". Le versioni di build elencate nella tabella non sono specificate nelle versioni autogestite elencate in GitHub. Aggiorna la versione completando le attività in una delle seguenti opzioni:

- Se non disponi di impostazioni personalizzate per il componente aggiuntivo, esegui il comando sotto l'`To apply this release:`intestazione GitHub relativa alla versione a cui stai eseguendo l'aggiornamento.
- Se disponi di impostazioni personalizzate, scarica il file manifesto con il comando seguente. Passa *`https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/master/aws-k8s-cni.yaml`* all'URL della versione a GitHub cui stai effettuando l'aggiornamento.

```
curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.3/config/
master/aws-k8s-cni.yaml
```

Se necessario, modifica il manifesto con le impostazioni personalizzate del backup che hai effettuato in una fase precedente, quindi applica il file modificato al tuo cluster. Se i tuoi nodi non hanno accesso ai repository privati Amazon ECR di Amazon EKS da cui sono estratte le immagini (consulta le righe che iniziano con `image:` nel manifesto), dovrai scaricare le immagini, copiarle nel tuo repository e modificare il manifesto per estrarre le immagini dal repository. Per ulteriori informazioni, consulta the section called "Copia di un'immagine in un repository".

```
kubectl apply -f aws-k8s-cni.yaml
```

4. Verifica che la nuova versione sia ora installata nel cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Di seguito viene riportato un output di esempio:

```
v1.20.3
```

5. (Facoltativo) Installa `cni-metrics-helper` sul tuo cluster. Raccoglie le informazioni sull'interfaccia di rete elastica e sull'indirizzo IP, le aggrega a livello di cluster e pubblica le metriche su Amazon. CloudWatch Per ulteriori informazioni, consulta on. cni-metrics-helper GitHub

## Configurare il plug-in CNI di Amazon VPC per l'utilizzo di IRSA

Il plug-in CNI di Amazon VPC per Kubernetes è il plug-in per reti Pod nei cluster Amazon EKS. Il plug-in è responsabile dell'allocazione degli indirizzi IP VPC ai Kubernetes Pods e della configurazione della rete necessaria per i Pod su ciascun nodo.

> **Note**
>
> Il plug-in CNI di Amazon VPC supporta anche Identità di Pod di Amazon EKS. Per ulteriori informazioni, consulta the section called "Assegnazione di un ruolo IAM".

Il plug-in:

- Richiede le autorizzazioni di AWS Identity and Access Management (IAM). Se il cluster utilizza la `IPv4` famiglia, le autorizzazioni sono specificate nella politica `AmazonEKS_CNI_Policy` AWS gestita. Se il cluster utilizza la famiglia `IPv6`, le autorizzazioni devono essere aggiunte a una policy IAM creata dall'utente; per le istruzioni, consulta the section called "`Crea una policy IAM per i cluster che utilizzano la famiglia IPv6`". È possibile collegare questa policy al ruolo del nodo IAM Amazon EKS o a un ruolo IAM separato. Per istruzioni su come allegare la policy al ruolo IAM del nodo Amazon EKS, consulta the section called "Ruolo IAM del nodo". Si consiglia di assegnarlo a un ruolo separato, come descritto in questo argomento.
- Crea ed è configurato per l'utilizzo di un account di servizio Kubernetes denominato `aws-node` quando viene implementato. L'account di servizio è associato a un `clusterrole` Kubernetes di nome `aws-node`, a cui vengono assegnate le autorizzazioni Kubernetes richieste.

> **Note**
>
> I Pod per il plug-in CNI di Amazon VPC per Kubernetes hanno accesso alle autorizzazioni assegnate al ruolo IAM del nodo Amazon EKS, a meno che non si blocchi l'accesso al servizio di metadati a IMDS. Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

- Richiede un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Richiede un provider AWS Identity and Access Management (IAM) OpenID Connect (OIDC) esistente per il cluster. Per determinare se si dispone già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Passaggio 1: Creazione di un ruolo IAM per il plug-in CNI di Amazon VPC per Kubernetes

1. Determinare la famiglia IP del cluster.

   ```
   aws eks describe-cluster --name my-cluster | grep ipFamily
   ```

   Di seguito viene riportato un output di esempio.

   ```
   "ipFamily": "ipv4"
   ```

   L'output potrebbe invece restituire `ipv6`.

2. Creare il ruolo IAM. Puoi usare `eksctl` or `kubectl` e la AWS CLI per creare il tuo ruolo IAM.

   eksctl

   - Creare un ruolo IAM e collegare ad esso la policy IAM con il comando corrispondente alla famiglia IP del cluster. Il comando crea e distribuisce uno AWS CloudFormation stack che crea un ruolo IAM, allega la policy specificata e annota l'account del servizio `aws-node` Kubernetes esistente con l'ARN del ruolo IAM creato.
     - `IPv4`

       Sostituire *`my-cluster`* con il proprio valore.

       ```
       eksctl create iamserviceaccount \
           --name aws-node \
           --namespace kube-system \
           --cluster my-cluster \
           --role-name AmazonEKSVPCCNIRole \
           --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
           --override-existing-serviceaccounts \
           --approve
       ```

     - `IPv6`

Sostituire *my-cluster* con il proprio valore. Sostituisci *111122223333* con il tuo ID account e *AmazonEKS_CNI_IPv6_Policy* con il nome della policy IPv6. Se non disponi di una policy IPv6, consulta the section called "Crea una policy IAM per i cluster che utilizzano la famiglia IPv6" per crearne una. Per utilizzare IPv6 con il cluster, questo deve soddisfare diversi requisiti. Per ulteriori informazioni, consulta the section called "IPv6".

```
eksctl create iamserviceaccount \
    --name aws-node \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKSVPCCNIRole \
    --attach-policy-arn arn:aws: iam::111122223333:policy/
AmazonEKS_CNI_IPv6_Policy \
    --override-existing-serviceaccounts \
    --approve
```

kubectl e la CLI AWS

i. Visualizzare l'URL del provider OIDC del cluster.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

Di seguito viene riportato un output di esempio.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

Se non viene restituito alcun output, devi creare un provider OIDC IAM per il cluster.

ii. Copia i seguenti contenuti in un file denominato *vpc-cni-trust-policy.json*. Sostituisci *111122223333* con il tuo ID account e *EXAMPLED539D4633E53DE1B71EXAMPLE* con l'output restituito nel passaggio precedente. Sostituisci *region-code* con la AWS regione in cui si trova il cluster.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
                }
            }
        }
    ]
}
```

iii. Crea il ruolo. Puoi sostituire *AmazonEKSVPCCNIRole* con un nome a tua scelta.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

iv. Allega la policy IAM richiesta al ruolo. Esegui il comando corrispondente alla famiglia IP del cluster.

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

- IPv6

  Sostituisci *111122223333* con il tuo ID account e *AmazonEKS_CNI_IPv6_Policy* con il nome della policy IPv6. Se non disponi di una policy IPv6, consulta the section called "Crea una policy IAM per i cluster che utilizzano la famiglia IPv6" per crearne una. Per utilizzare IPv6 con il cluster, questo deve soddisfare diversi requisiti. Per ulteriori informazioni, consulta the section called "IPv6".

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
```

```
    --role-name AmazonEKSVPCCNIRole
```

v. Esegui il comando seguente per annotare l'account di servizio `aws-node` con l'ARN del ruolo IAM creato in precedenza. Sostituire i valori di esempio con i propri valori.

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-arn=arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole
```

3. (Facoltativo) Configura il tipo di endpoint del servizio AWS Security Token utilizzato dal tuo account del servizio Kubernetes. Per ulteriori informazioni, consulta the section called "Endpoint STS".

Fase 2: Plugin Re-deploy Amazon VPC CNI per Kubernetes Pods

1. Eliminare e ricreare i Pod esistenti associati all'account di servizio per applicare le variabili di ambiente delle credenziali. L'annotazione non viene applicata ai Pod attualmente in esecuzione senza l'annotazione. Il comando seguente elimina i `aws-node` DaemonSet Pod esistenti e li distribuisce con l'annotazione dell'account di servizio.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Confermare che tutti i Pod sono stati riavviati.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Descrivere uno dei Pod e verificare che le variabili di ambiente `AWS_WEB_IDENTITY_TOKEN_FILE` e `AWS_ROLE_ARN` esistano. Sostituiscilo *`cpjw7`* con il nome di uno dei tuoi Pod restituito nell'output del passaggio precedente.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

Di seguito viene riportato un output di esempio.

```
AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
```

```
      AWS_WEB_IDENTITY_TOKEN_FILE:           /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

Vengono restituiti due set di risultati duplicati perché il Pod contiene due container. I due container hanno gli stessi valori.

Se il Pod utilizza l'endpoint AWS regionale, nell'output precedente viene restituita anche la riga seguente.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Passaggio 3: rimozione della policy CNI dal ruolo IAM del nodo

Se al tuo ruolo IAM del nodo Amazon EKS è attualmente associata la policy `AmazonEKS_CNI_Policy` IAM (`IPv4`) o una policy IPv6 e hai creato un ruolo IAM separato, collegato invece la policy ad esso e l'hai assegnata all'account di servizio `aws-node` Kubernetes, ti consigliamo di rimuovere la policy dal ruolo del nodo con il comando AWS CLI che corrisponde alla famiglia IP del tuo cluster. Sostituisci *`AmazonEKSNodeRole`* con il nome del ruolo del nodo.

- `IPv4`

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::aws:policy/AmazonEKS_CNI_Policy
```

- `IPv6`

  Sostituisci *`111122223333`* con il tuo ID account e *`AmazonEKS_CNI_IPv6_Policy`* con il nome della policy `IPv6`.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

### Crea una policy IAM per i cluster che utilizzano la famiglia IPv6

Se hai creato un cluster che utilizza la famiglia `IPv6` e sul cluster è configurata la versione `1.10.1` o successiva del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes, devi creare una policy IAM da assegnare a un ruolo IAM. Se disponi di un cluster che non è stato configurato con

la famiglia `IPv6` al momento della creazione, prima di utilizzare `IPv6`, devi creare un nuovo cluster. Per ulteriori informazioni sull'uso di `IPv6` con il cluster, consulta the section called "IPv6".

1. Copia il testo seguente e salvalo in un file denominato `vpc-cni-ipv6-policy.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. Creare la policy IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

## Ulteriori informazioni sulle modalità e sulla configurazione di CNI di VPC

Il plugin CNI di Amazon VPC fornisce una rete per pod. Utilizza la tabella seguente per ulteriori informazioni sulle funzionalità di rete disponibili.

| Caratteristica di rete | Ulteriori informazioni |
| --- | --- |
| Configura il cluster per assegnare indirizzi IPv6 a cluster, pod e servizi | the section called "IPv6" |
| Utilizza la traduzione degli indirizzi di rete di origine IPv4 per i pod | the section called "Traffico in uscita" |
| Limita il traffico di rete verso e dai pods | the section called "Limitare il traffico" |
| Configura la selezione della sottorete per gli indirizzi IP del Pod | the section called "Selezione della sottorete" |
| Personalizza l'interfaccia di rete secondaria nei nodi | the section called "Rete personalizzata" |
| Aumenta gli indirizzi IP per il nodo | the section called "Aumentare gli indirizzi IP" |
| Utilizza gruppi di sicurezza per il traffico di rete dei pod | the section called "Gruppi di sicurezza per pod" |
| Utilizza interfacce di rete multiple per pod | the section called "Interfacce multiple" |

Informazioni sugli indirizzi IPv6 per cluster, pod e servizi

Si applica a: Pod con istanze Amazon EC2 e Pod Fargate

Per impostazione predefinita, Kubernetes assegna indirizzi `IPv4` a pod e servizi. Invece di assegnare indirizzi `IPv4` a pod e servizi, è possibile configurare il cluster in modo da assegnare loro indirizzi `IPv6`. Amazon EKS non supporta pod o servizi con doppio stack, anche se Kubernetes lo fa. Di conseguenza, non è possibile assegnare sia indirizzi `IPv4` che `IPv6` a pod e servizi.

È possibile selezionare la famiglia IP che si desidera utilizzare per il cluster al momento della creazione. Non sarà possibile modificare la famiglia dopo aver creato il cluster.

Per un tutorial sull'implementazione di un cluster `IPv6` di Amazon EKS, consultare the section called "Implementazione".

Di seguito sono riportate alcune considerazioni sull'utilizzo della funzionalità:

### Supporto delle funzionalità IPv6

- Nessun supporto per Windows: i Pod e i servizi Windows non sono supportati.
- Nitro-based Nodi EC2 richiesti: è possibile utilizzarli solo `IPv6` con nodi AWS Nitro-based Amazon EC2 o Fargate.
- Nodi EC2 e Fargate supportati: è possibile utilizzare `IPv6` con the section called "Gruppi di sicurezza per pod" con nodi Amazon EC2 e nodi Fargate.
- Outpost non supportati: non è possibile usare `IPv6` con Amazon EKS su AWS Outposts.
- FSx per Lustre non è supportato: il the section called "Amazon FSx per Lustre" non è supportato.
- Rete personalizzata non supportata: per ridurre l'esaurimento dell'indirizzo IP, è possibile utilizzare the section called "Rete personalizzata" anche se in precedenza è stato usato `IPv6`. Non è possibile utilizzare una rete personalizzata con `IPv6`. Se si utilizza una rete personalizzata per l'isolamento della rete, potrebbe essere necessario continuare a utilizzare la rete personalizzata e la famiglia `IPv4` per i cluster.

Assegnazioni dell'indirizzo IP

- Servizi Kubernetes: ai servizi Kubernetes vengono assegnati solo indirizzi. `IPv6` Non vengono assegnati a indirizzi IPv4.
- Pod: ai pod viene assegnato un indirizzo IPv6 e un indirizzo IPv4 locale dell'host. L'indirizzo IPv4 locale dell'host viene assegnato utilizzando un plug-in CNI locale dell'host concatenato con CNI di VPC e l'indirizzo non viene segnalato al piano di controllo di Kubernetes. Viene utilizzato solo quando un pod deve comunicare con una risorsa IPv4 esterna in un altro Amazon VPC o in Internet. L'indirizzo IPv4 locale dell'host viene sottoposto a SNAT (tramite CNI di VPC) all'indirizzo IPv4 primario dell'ENI primario del nodo worker.
- Pod e servizi: i pod e i servizi ricevono solo indirizzi `IPv6`, non quelli `IPv4`. Quando i Pod devono comunicare con endpoint `IPv4` esterni, utilizzano il NAT sul nodo stesso. Questa funzionalità NAT integrata elimina la necessità di DNS64 e NAT64. Per il traffico che richiede un accesso pubblico a Internet, il traffico del pod è l'indirizzo di rete di origine tradotto in un indirizzo IP pubblico.
- Indirizzi di instradamento: quando un Pod comunica all'esterno del VPC, il suo indirizzo `IPv6` originale viene preservato (non tradotto nell'indirizzo `IPv6` del nodo). Questo traffico viene instradato direttamente attraverso un gateway Internet o gateway Internet egress-only.
- Nodi: a tutti i nodi viene assegnato un indirizzo `IPv4` e `IPv6`.

- Pod Fargate: ogni Pod Fargate riceve un indirizzo IPv6 dal CIDR specificato per la sottorete in cui è stato implementato. L'unità hardware sottostante che esegue i pod Fargate ottiene un indirizzo IPv4 e IPv6 univoci dai CIDR assegnati alla sottorete in cui è implementata l'unità hardware.

### Come usare **IPv6 con EKS**

- Creare un nuovo cluster: è necessario creare un nuovo cluster e specificare che si vuole utilizzare la famiglia IPv6 per quel cluster. Non è possibile abilitare la famiglia IPv6 per un cluster aggiornato da una versione precedente. Per istruzioni su come creare un nuovo cluster, consultare the section called "Implementazione".
- Usare un CNI di VPC recente: implementare la versione CNI di Amazon VPC o successiva 1.10.1. Per impostazione predefinita, viene distribuita questa versione o una versione successiva. Dopo aver implementato il componente aggiuntivo, non sarà possibile eseguire il downgrade del componente aggiuntivo CNI di Amazon VPC a una versione inferiore a 1.10.1 senza rimuovere tutti i nodi in tutti i gruppi di nodi nel cluster.
- Configurare CNI di VPC per **IPv6**: se si utilizzano i nodi Amazon EC2, è necessario configurare il componente aggiuntivo CNI di Amazon VPC con delega del prefisso IP e IPv6. Se durante la creazione del cluster si sceglie la famiglia IPv6, la versione 1.10.1 del componente aggiuntivo utilizza per impostazione predefinita questa configurazione. Questo è il caso di un componente aggiuntivo autogestito o Amazon EKS. Per ulteriori informazioni sulla delega dei prefissi IP, consultare the section called "Aumentare gli indirizzi IP".
- Configurare indirizzi **IPv4** e **IPv6**: quando si crea un cluster, il VPC e le sottoreti specificate devono avere un intervallo CIDR IPv6 assegnato al VPC e alle sottoreti specificate. Devono inoltre disporre di un blocco CIDR IPv4 assegnato. Questo perché, anche se si desidera utilizzare solo IPv6, il funzionamento di un VPC richiede comunque un blocco CIDR IPv4. Per ulteriori informazioni, consulta Associazione di un blocco CIDR IPv6 al VPC nella Guida per l'utente di Amazon VPC.
- Auto-assign Indirizzi IPv6 ai nodi:  quando si creano i nodi, è necessario specificare le sottoreti configurate per l'assegnazione automatica degli indirizzi. IPv6 In caso contrario, non sarà possibile implementare i nodi. Per impostazione predefinita, questa configurazione è disabilitata. Per ulteriori informazioni, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv6 pubblici della sottorete nella Guida per l'utente di Amazon VPC.
- Impostare le tabelle di routing per usare **IPv6**: le tabelle di routing assegnate alle sottoreti devono avere percorsi per gli indirizzi IPv6. Per ulteriori informazioni, consulta Migrazione a IPv6 nella Guida per l'utente di Amazon VPC.

- Impostare gruppi di sicurezza per **IPv6**: i gruppi di sicurezza devono consentire gli indirizzi IPv6. Per ulteriori informazioni, consulta Migrazione a IPv6 nella Guida per l'utente di Amazon VPC.
- Configura il load balancer: utilizza la versione 2.3.1 o successiva del Load Balancer Controller per bilanciare il AWS carico delle applicazioni HTTP utilizzando il the section called "Bilanciamento del carico di applicazione" o il traffico di rete utilizzando i the section called "Bilanciamento del carico di rete" to IPv6 Pods con uno dei due load balancer in modalità IP, ma non in modalità istanza. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer".
- Aggiungere una policy IAM **IPv6**: è necessario allegare una policy IAM IPv6 al ruolo IAM del nodo o del CNI. Tra i due, consigliamo di allegarlo a un ruolo IAM CNI. Per ulteriori informazioni, consulta the section called "Crea una policy IAM per i cluster che utilizzano la famiglia IPv6" e the section called "Passaggio 1: Creazione di un ruolo IAM per il plug-in CNI di Amazon VPC per Kubernetes".
- Valuta tutti i componenti: esegui una valutazione approfondita delle applicazioni, dei componenti aggiuntivi di Amazon EKS e dei AWS servizi con cui ti integri prima di distribuire i cluster. IPv6 Questo per garantire che tutto funzioni come previsto con IPv6.

Implementazione di un cluster Amazon EKS **IPv6** e nodi Amazon Linux gestiti

In questo tutorial, implementi un Amazon VPC IPv6, un cluster Amazon EKS con la famiglia IPv6 e un gruppo di nodi gestiti con nodi Amazon Linux per Amazon EC2. Non è possibile implementare nodi Windows per Amazon EC2 in un cluster IPv6. È inoltre possibile implementare nodi Fargate nel cluster, sebbene tali istruzioni non siano fornite in questo argomento per semplicità.

Prerequisiti

Prima di iniziare questo tutorial, completare le seguenti operazioni:

Installare e configurare gli strumenti e le risorse elencati di seguito, che sono necessari per creare e gestire un cluster Amazon EKS.

- # consigliabile acquisire familiarità con tutte le impostazioni e implementare un cluster con le impostazioni che soddisfano i requisiti specifici. Per ulteriori informazioni, consultare the section called "Creazione di un cluster ", the section called "Gruppi di nodi gestiti" e le considerazioni per questo argomento. Alcune impostazioni possono essere abilitate solo quando viene creato il cluster.
- Lo strumento a riga di comando kubectl è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di

Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

- Il responsabile della sicurezza IAM che stai utilizzando deve disporre delle autorizzazioni per lavorare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi AWS CloudFormation, un VPC e le risorse correlate. Per ulteriori informazioni, consultare le pagine Actions e Using service-linked roles nella Guida per l'utente di IAM.
- Se lo utilizzi`eksctl`, installa la versione `0.215.0` o successiva sul tuo computer. Per installarlo o aggiornarlo, consulta la sezione Installation nella documentazione di `eksctl`.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum`apt-get, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente. Se si utilizza la AWS CloudShell, potrebbe essere necessario installare la versione 2.12.3 o successiva o 1.27.160 o successiva della CLI AWS , poiché la versione CLI predefinita installata in potrebbe essere una versione precedente. AWS AWS CloudShell

È possibile utilizzare eksctl o la CLI per implementare un cluster `IPv6`.

Implementare un cluster IPv6 con eksctl

a. Crea il file `ipv6-cluster.yaml`. Copia il comando seguente sul tuo dispositivo. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituisci *`my-cluster`* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e dell' AWS account in cui stai creando il cluster.
   - Sostituisci *`region-code`* con qualsiasi AWS regione supportata da Amazon EKS. Per un elenco di AWS regioni, consulta gli endpoint e le quote di Amazon EKS nella guida di riferimento AWS generale.

- Il valore per `version` con la versione del tuo cluster. Per ulteriori informazioni, consultare la pagina Amazon EKS supported versions.
- Sostituisci *`my-nodegroup`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti.
- Sostituisci *`t3.medium`* con qualsiasi tipo di istanza di AWS Nitro System.

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

b. Creare il cluster.

```
eksctl create cluster -f ipv6-cluster.yaml
```

La creazione di cluster richiede diversi minuti. Non procedere fino a quando non viene visualizzata l'ultima riga di output, simile alla seguente.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

c. Confermare che ai pod predefiniti siano assegnati indirizzi IPv6.

```
kubectl get pods -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                     NODE                                              NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

d. Confermare che ai servizi di default siano assegnati indirizzi IPv6.

```
kubectl get services -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

e. (Facoltativo) Implementare un'applicazione di esempio o implementare AWS Load Balancer Controller e un'applicazione di esempio per eseguire il bilanciamento del carico delle applicazioni HTTP con the section called "Bilanciamento del carico di applicazione" o del traffico di rete con the section called "Bilanciamento del carico di rete" verso i pod IPv6.

f. Una volta terminato con il cluster e i nodi creati per questo tutorial, è necessario eliminare le risorse create con il seguente comando.

```
eksctl delete cluster my-cluster
```

Implementa un cluster IPv6 con AWS CLI

**Important**

- È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:

  ```
  aws sts get-caller-identity
  ```

- È necessario completare tutti i passaggi di questa procedura nella stessa shell. Diversi passaggi utilizzano le variabili impostate nelle fasi precedenti. Le fasi che utilizzano le variabili non funzioneranno correttamente se i relativi valori sono impostati in una shell diversa. Se utilizzate il AWS CloudShellper completare la procedura seguente, ricordate che se non interagite con esso utilizzando la tastiera o il puntatore per circa 20-30 minuti, la sessione di shell termina. I processi in esecuzione non sono considerati interazioni.
- Le istruzioni sono scritte per la shell Bash e per altre shell potrebbero essere necessarie ulteriori regolazioni.

Sostituire tutti i valori di esempio nelle fasi di questa procedura con i propri valori.

a. Emettere i seguenti comandi per impostare alcune variabili utilizzate nelle fasi successive. Sostituiscilo *region-code* con la AWS regione in cui desideri distribuire le risorse. Il valore può essere qualsiasi AWS regione supportata da Amazon EKS. Per un elenco di AWS regioni, consulta gli endpoint e le quote di Amazon EKS nella guida di riferimento AWS generale. Sostituisci *my-cluster* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster. Sostituisci *my-nodegroup* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti. Sostituisci *111122223333* con l'ID del tuo account.

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

b. Creare un Amazon VPC con sottoreti pubbliche e private che soddisfino i requisiti Amazon EKS e IPv6.

   i. Eseguite il comando seguente per impostare una variabile per il nome dello stack. AWS CloudFormation Puoi sostituire *my-eks-ipv6-vpc* con un nome a tua scelta.

   ```
   export vpc_stack_name=my-eks-ipv6-vpc
   ```

   ii. Crea un IPv6 VPC utilizzando un AWS CloudFormation modello.

   ```
   aws cloudformation create-stack --region $region_code --stack-name $vpc_stack_name
    \
     --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
   cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
   ```

   La creazione dello stack richiede alcuni minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva finché l'output del comando non è CREATE_COMPLETE.

   ```
   aws cloudformation describe-stacks --region $region_code --stack-name
    $vpc_stack_name --query Stacks[].StackStatus --output text
   ```

   iii. Recuperare gli ID delle sottoreti pubbliche che sono state create.

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text
```

Di seguito viene riportato un output di esempio:

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

iv. Abilitare l'opzione di assegnazione automatica dell'indirizzo IPv6 per le sottoreti pubbliche create.

```
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

v. Recupera i nomi delle sottoreti e dei gruppi di sicurezza creati dal modello dallo AWS CloudFormation stack distribuito e memorizzali in variabili da utilizzare in un passaggio successivo.

```
security_groups=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --output
 text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --stack-
name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --output
 text)

subnets=${public_subnets},${private_subnets}
```

c. Crea un ruolo IAM del cluster e collegalo alla policy gestita IAM richiesta dall'AMI per Amazon EKS. I cluster Kubernetes gestiti da Amazon EKS effettuano chiamate ad altri AWS servizi per tuo conto per gestire le risorse che utilizzi con il servizio.

   i. Per creare il file `eks-cluster-role-trust-policy.json`, emetti il seguente comando:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   ii. Eseguire questo comando per impostare una variabile per il nome del ruolo. Puoi sostituire *myAmazonEKSClusterRole* con un nome a tua scelta.

```
export cluster_role_name=myAmazonEKSClusterRole
```

   iii. Crea il ruolo.

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

   iv. Recuperare l'ARN del ruolo IAM e memorizzarlo in una variabile per una fase successiva.

```
CLUSTER_IAM_ROLE=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

   v. Allega la policy IAM gestita da Amazon EKS richiesta al ruolo.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

d. Creare il cluster.

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $CLUSTER_IAM_ROLE --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

i. NOTA: è possibile ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

   La creazione del cluster richiede diversi minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva fino a quando l'output del comando non è `ACTIVE`.

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

e. Creare o aggiornare un file `kubeconfig` per il cluster in modo che sia possibile comunicare con il cluster.

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

   Per impostazione predefinita, il file `config` viene creato in `~/.kube` o la configurazione del nuovo cluster viene aggiunta a un file `config` esistente in `~/.kube`.

f. Creare un ruolo IAM del nodo.

   i. Per creare il file `vpc-cni-ipv6-policy.json`, emetti il seguente comando:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
```

```
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

ii. Creare la policy IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

iii. Per creare il file `node-role-trust-relationship.json`, emetti il seguente comando:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

iv. Eseguire questo comando per impostare una variabile per il nome del ruolo. Puoi sostituire *`AmazonEKSNodeRole`* con un nome a tua scelta.

```
export node_role_name=AmazonEKSNodeRole
```

v. Creare il ruolo IAM.

```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
 file://"node-role-trust-relationship.json"
```

vi. Allegare la policy IAM al ruolo IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> Per semplicità, in questo tutorial la policy è allegata a questo ruolo IAM. Tuttavia, in un cluster di produzione, si consiglia di allegare la policy a un ruolo IAM separato. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

vii. Allegare al ruolo IAM le due policy gestite IAM richieste.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

viii Recuperare l'ARN del ruolo IAM e memorizzarlo in una variabile per una fase successiva.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn" --
output text)
```

g. Creare un gruppo di nodi gestito.

i. Visualizzare gli ID delle sottoreti creati in una fase precedente.

```
echo $subnets
```

Di seguito viene riportato un output di esempio:

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-0377963d69EXAMPLE,subnet-0c05f819
```

ii. Creare il gruppo di nodi. Sostituisci *0a1a56c486EXAMPLE*, *099e6ca77aEXAMPLE*, *0377963d69EXAMPLE*, and *0c05f819d5EXAMPLE* con i valori restituiti nell'output della fase precedente. Assicurarsi di rimuovere le virgole tra gli ID delle sottoreti dall'output precedente nel comando seguente. Puoi sostituire *t3.medium* con qualsiasi tipo di istanza di AWS Nitro System.

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

La creazione del gruppo di nodi richiede alcuni minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

h. Confermare che ai pod predefiniti siano assegnati indirizzi `IPv6` nella colonna `IP`.

```
kubectl get pods -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                    NODE                                            NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

```
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

i. Confermare che ai servizi di default siano assegnati gli indirizzi IPv6 nella colonna IP.

```
kubectl get services -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

j. (Facoltativo) Implementare un'applicazione di esempio o implementare AWS Load Balancer Controller e un'applicazione di esempio per eseguire il bilanciamento del carico delle applicazioni HTTP con the section called "Bilanciamento del carico di applicazione" o del traffico di rete con the section called "Bilanciamento del carico di rete" verso i pod IPv6.

k. Una volta terminato con il cluster e i nodi creati per questo tutorial, è necessario eliminare le risorse create con i seguenti comandi. Assicurarsi di non utilizzare nessuna delle risorse al di fuori di questo tutorial prima di eliminarle.

i. Se si sta completando questo passaggio in una shell diversa da quella in cui sono state completate le fasi precedenti, impostare i valori di tutte le variabili utilizzate nelle fasi precedenti sostituendo i valori di esempio con quelli specificate durante le fasi precedenti. Se si sta completando questo passaggio nella stessa shell in cui sono state completate le fasi precedenti, andare alla fase successiva.

```
export region_code=region-code
export vpc_stack_name=my-eks-ipv6-vpc
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
export node_role_name=AmazonEKSNodeRole
export cluster_role_name=myAmazonEKSClusterRole
```

ii. Eliminare il gruppo di nodi.

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name
```

L'eliminazione richiede pochi minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva se viene restituito un output.

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --query
 nodegroups --output text
```

iii. Elimina il cluster.

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

L'eliminazione del cluster richiede alcuni minuti. Prima di continuare, accertarsi che il cluster sia eliminato con il comando seguente:

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

Non procedere alla fase successiva finché l'output non è simile al seguente:

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

iv. Eliminare le risorse IAM create. Sostituisci *AmazonEKS_CNI_IPv6_Policy* con il nome scelto, se è stato scelto un nome diverso da quello usato nei passaggi precedenti.

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSClusterPolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws: iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

v. Elimina lo AWS CloudFormation stack che ha creato il VPC.

```
aws cloudformation delete-stack --region $region_code --stack-name $vpc_stack_name
```

Abilitare l'accesso a Internet in uscita per i pod

Si applica a: nodi Fargate Linux `IPv4`, nodi Linux con istanze Amazon EC2

Le informazioni contenute in questa sezione non sono applicabili al cluster se l'implementazione è avvenuta tramite la famiglia `IPv6`, in quanto gli indirizzi `IPv6` non sono convertiti in rete. Per ulteriori informazioni sull'uso di `IPv6` con il cluster, consulta the section called "IPv6".

Per impostazione predefinita, a ciascun pod nel cluster viene assegnato un indirizzo `IPv4` privato da un routing interdominio senza classi (CIDR) associato al VPC in cui è implementato il pod. I pod nello stesso VPC comunicano tra loro utilizzando questi indirizzi IP privati come endpoint. Quando un pod comunica a un qualsiasi indirizzo `IPv4` che non è incluso in un intervallo CIDR associato al VPC, il plugin CNI di Amazon VPC (per Linux o Windows) converte l'indirizzo `IPv4` del pod nell'indirizzo `IPv4` privato principale dell'interfaccia di rete elastica principale del nodo su cui è in esecuzione il pod, ovvero * per impostazione predefinita.

> **Note**
>
> Per i nodi Windows, ci sono ulteriori dettagli da considerare. Per impostazione predefinita, il plugin CNI VPC per Windows è definito con una configurazione di rete in cui il traffico verso una destinazione all'interno dello stesso VPC è escluso per SNAT. Ciò significa che la comunicazione VPC interna ha SNAT disabilitata e l'indirizzo IP assegnato al pod è indirizzabile all'interno del VPC. Tuttavia, il traffico verso una destinazione esterna al VPC ha l'IP di origine del pod tradotto tramite SNAT nell'indirizzo IP primario dell'interfaccia di rete elastica (ENI) dell'istanza. Questa configurazione predefinita per Windows garantisce che il pod possa accedere alle reti esterne al VPC allo stesso modo dell'istanza host.

A causa di questo comportamento:

- I tuoi Pod possono comunicare con le risorse Internet se il nodo su cui sono in esecuzione ha un indirizzo IP pubblico o elastico e si trova in una sottorete pubblica la cui tabella di routing ha un percorso verso un gateway https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html Internet o si trova in una sottorete privata la cui tabella di routing ha

un percorso verso un gateway NAT. Per questo motivo, ti consigliamo di implementare i nodi nelle sottoreti private, quando possibile.

- Per le versioni del plugin precedenti alla `1.8.0`, le risorse che si trovano nelle reti o nei VPC collegati al VPC del cluster tramite peering VPC, VPC di transito o AWS Direct Connect non possono avviare una comunicazione con i pod oltre le interfacce di rete elastiche secondarie. I pod possono tuttavia avviare comunicazioni con tali risorse e ricevere risposte.

Se una delle seguenti affermazioni è vera nel tuo ambiente, modifica la configurazione predefinita con il comando che segue.

- Si dispone di risorse nelle reti o nei VPC collegati al VPC del cluster tramite peering VPC, VPC di transito o AWS Direct Connect che devono avviare la comunicazione con i pod tramite un indirizzo `IPv4`, con la versione del plugin precedente alla `1.8.0`.
- Vuoi che l'indirizzo IP di origine del Pod venga preservato quando il traffico lascia il nodo, con SNAT eseguito da un gateway NAT anziché dal nodo, ad esempio, in modo che le appliance o i log di rete osservino l'indirizzo IP del Pod. I tuoi nodi devono trovarsi in una sottorete privata con un percorso verso un gateway NAT, perché il Pod-sourced traffico non può passare attraverso un gateway Internet.

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> Le variabili di configurazione `AWS_VPC_K8S_CNI_EXTERNALSNAT` e `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` di CNI non sono applicabili ai nodi Windows. La disabilitazione di SNAT non è supportata per Windows. Per quanto riguarda l'esclusione di un elenco di CIDR `IPv4` da SNAT, è possibile definirla specificando il parametro `ExcludedSnatCIDRs` nello script di bootstrap di Windows. Per ulteriori informazioni su questo parametro, consulta la sezione the section called "Parametri di configurazione dello script di bootstrap".

Rete host

*Se la specifica di un pod contiene `hostNetwork=true` (il valore predefinito è `false`), il suo indirizzo IP non viene convertito in un indirizzo diverso. Per impostazione predefinita, questo

comportamento è presente per i pod di `kube-proxy` e del plugin CNI di Amazon VPC per Kubernetes in esecuzione sul cluster. Per questi pod, l'indirizzo IP corrisponde a quello principale del nodo, poiché l'indirizzo IP del pod non viene convertito. Per ulteriori informazioni sulle `hostNetwork` impostazioni di un Pod, consulta PodSpec v1 core nel riferimento alle API Kubernetes.

Limita il traffico di Pod con le policy di rete di Kubernetes

Panoramica di

Per impostazione predefinita, in Kubernetes non ci sono restrizioni per indirizzi IP, porte o connessioni tra qualsiasi Pod nel cluster o tra i pod e le risorse in qualsiasi altra rete. È possibile utilizzare la policy di rete di Kubernetes per limitare il traffico di rete da e verso i pod. Per ulteriori informazioni, consulta Network Policies nella documentazione Kubernetes.

Politica di rete standard

È possibile utilizzare lo standard `NetworkPolicy` per segmentare il traffico da pod a pod nel cluster. Queste policy di rete operano ai livelli 3 e 4 del modello di rete OSI, consentendoti di controllare il flusso di traffico a livello di indirizzo IP o porta all'interno del tuo cluster Amazon EKS. Le politiche di rete standard sono limitate a livello di namespace.

Casi d'uso

- Segmenta il traffico di rete tra i carichi di lavoro per garantire che solo le applicazioni correlate possano comunicare tra loro.
- Isola i tenant a livello di namespace utilizzando politiche per imporre la separazione della rete.

Esempio

Nella politica riportata di seguito, il traffico in uscita dai  pod delle webapp nel namespace sun è limitato.

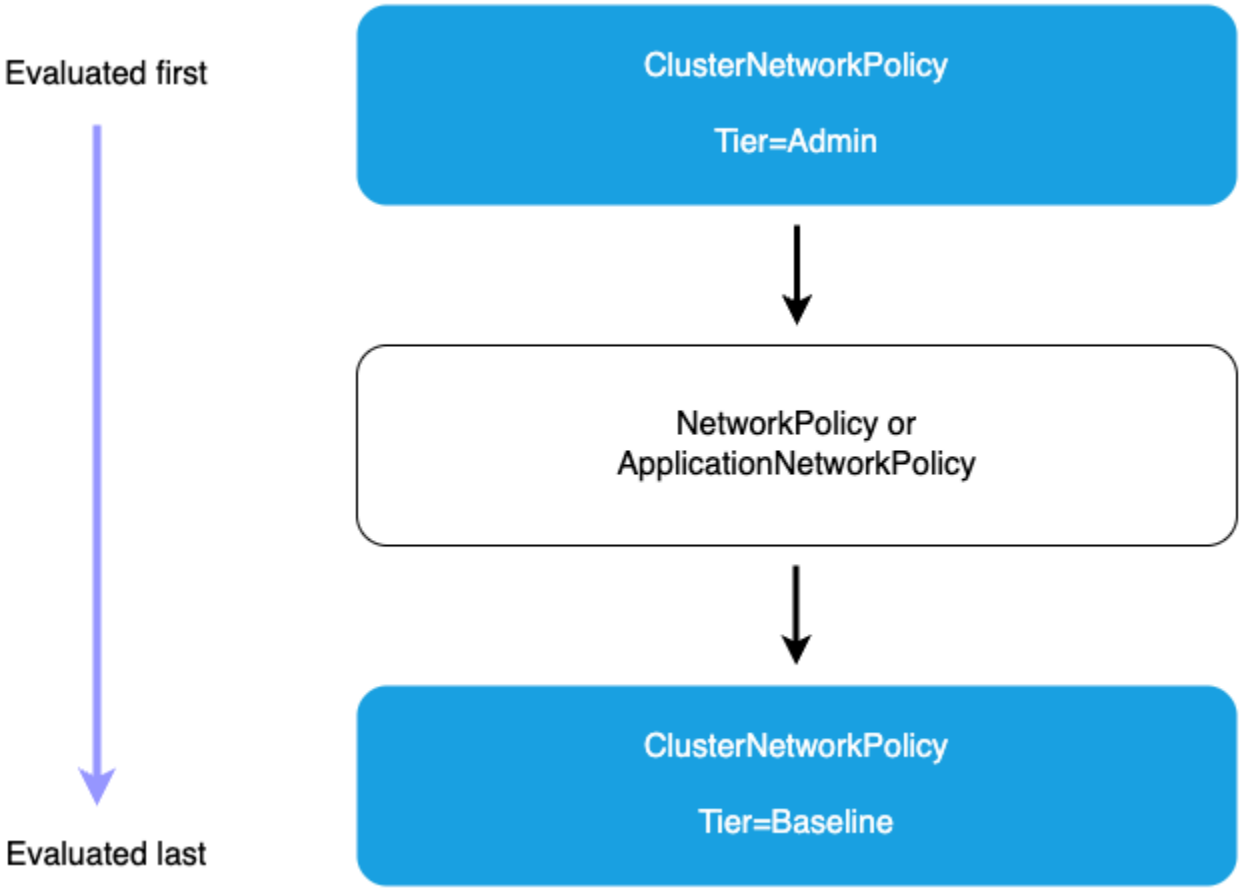
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: webapp-egress-policy
  namespace: sun
spec:
  podSelector:
    matchLabels:
```

```
      role: webapp
  policyTypes:
  - Egress
  egress:
  - to:
    - namespaceSelector:
        matchLabels:
          name: moon
      podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 8080
  - to:
    - namespaceSelector:
        matchLabels:
          name: stars
      podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 8080
```

La politica si applica ai pod con l'etichetta nel namespace. `role: webapp` `sun`

- Traffico consentito: pod con l'etichetta `role: frontend` nel namespace sulla porta TCP `moon` `8080`
- Traffico consentito: pod con l'etichetta `role: frontend` nello spazio dei nomi sulla porta TCP `stars` `8080`
- Traffico bloccato: tutto il resto del traffico in uscita dai pod viene negato implicitamente `webapp`

Politica di rete dell'amministratore (o del cluster)




È possibile utilizzare il `ClusterNetworkPolicy` per applicare uno standard di sicurezza di rete che si applica all'intero cluster. Invece di definire e mantenere ripetutamente una policy distinta per ogni namespace, è possibile utilizzare un'unica policy per gestire centralmente i controlli di accesso alla rete per i diversi carichi di lavoro nel cluster, indipendentemente dal rispettivo namespace.

Casi d'uso

- Gestisci centralmente i controlli di accesso alla rete per tutti (o un sottoinsieme di) carichi di lavoro nel tuo cluster EKS.
- Definisci un livello di sicurezza di rete predefinito in tutto il cluster.
- Estendi gli standard di sicurezza organizzativi all'ambito del cluster in un modo più efficiente dal punto di vista operativo.

### Esempio

Nella policy riportata di seguito, è possibile bloccare esplicitamente il traffico del cluster proveniente da altri namespace per impedire l'accesso di rete a un namespace sensibile dei carichi di lavoro.

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ClusterNetworkPolicy
metadata:
  name: protect-sensitive-workload
spec:
  tier: Admin
  priority: 10
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: earth
  ingress:
    - action: Deny
      from:
      - namespaces:
          matchLabels: {} # Match all namespaces.
      name: select-all-deny-all
```

### Note importanti

Le policy di rete nel plug-in Amazon VPC CNI per Kubernetes sono supportate nelle configurazioni elencate di seguito.

- Versione 1.21.0 (o successiva) del plug-in Amazon VPC CNI per le policy di rete standard e amministrative.
- Cluster configurato per indirizzi `IPv4` o `IPv6`.
- È possibile utilizzare le policy di rete con i gruppi di sicurezza per pod. Con le policy di rete, è possibile controllare tutte le comunicazioni all'interno del cluster. Con i gruppi di sicurezza per Pods, puoi controllare l'accesso ai AWS servizi dalle applicazioni all'interno di un Pod.
- È possibile utilizzare le policy di rete con le reti personalizzate e la delega del prefisso.

### Considerazioni

#### Architettura

- Quando si applicano le policy di rete del plug-in CNI di Amazon VPC per Kubernetes al cluster con il plug-in CNI di Amazon VPC per Kubernetes, è possibile applicare le policy solo ai nodi Amazon EC2 Linux. Non è possibile applicare le policy ai nodi Fargate o Windows.
- Le politiche di rete si applicano solo a uno `IPv4` `IPv6` degli indirizzi, ma non a entrambi. In un `IPv4` cluster, il VPC CNI assegna un `IPv4` indirizzo ai pod e applica le policy. `IPv4` In un `IPv6` cluster, il VPC CNI assegna un `IPv6` indirizzo ai pod e applica le policy. `IPv6` Tutte le regole di policy di rete `IPv4` applicate a un cluster `IPv6` vengono ignorate. Tutte le regole di policy di rete `IPv6` applicate a un cluster `IPv4` vengono ignorate.

Policy di rete

- Amazon EKS ottimizza l'applicazione delle policy di rete per i pod con un set di campi. `metadata.ownerReferences` Ciò include i pod gestiti da controller come Deployments, StatefulSets, DaemonSets Jobs e. CronJobs I pod autonomi creati direttamente senza un controller non sono `metadata.ownerReferences` impostati e l'applicazione delle policy di rete potrebbe non funzionare in modo affidabile per questi pod.
- È possibile applicare più policy di rete allo stesso pod. Quando sono configurate due o più policy che selezionano lo stesso pod, al pod vengono applicate tutte le policy.
- Il numero massimo di combinazioni di porte e protocolli per un singolo intervallo di indirizzi IP (CIDR) è 24 per tutte le policy di rete. Selettori come `namespaceSelector` risolvono verso uno o più CIDR. Se più selettori si risolvono in un unico CIDR o si specifica lo stesso CIDR diretto più volte nella stessa policy di rete o in diverse policy di rete, tutti questi fattori vengono conteggiati ai fini di questo limite.
- Per ognuno dei servizi di Kubernetes, la porta del servizio deve essere uguale a quella del container. Se si utilizzano porte con un nome assegnato, è necessario utilizzare lo stesso nome anche nelle specifiche del servizio.

Politiche di rete di amministrazione

1. Criteri del livello amministrativo (valutati per primi): tutti i livelli di amministrazione ClusterNetworkPolicies vengono valutati prima di qualsiasi altra politica. All'interno del livello Amministratore, le politiche vengono elaborate in ordine di priorità (prima il numero con la priorità più bassa). Il tipo di azione determina cosa succede dopo.
   - Azione di rifiuto (precedenza massima): quando una politica di amministrazione con un'azione Nega corrisponde al traffico, tale traffico viene immediatamente bloccato indipendentemente

da qualsiasi altra politica. Non vengono elaborate ClusterNetworkPolicy ulteriori NetworkPolicy regole. Ciò garantisce che i controlli di sicurezza a livello di organizzazione non possano essere sostituiti da policy a livello di namespace.

- Consenti azione: dopo aver valutato le regole di rifiuto, le politiche di amministrazione con azioni Consenti vengono elaborate in ordine di priorità (prima il numero con la priorità più bassa). Quando un'azione Consenti corrisponde, il traffico viene accettato e non viene effettuata alcuna ulteriore valutazione della policy. Queste policy possono concedere l'accesso a più namespace in base a selettori di etichette, fornendo un controllo centralizzato su quali carichi di lavoro possono accedere a risorse specifiche.
- Passare all'azione: le azioni passate nelle policy di livello amministrativo delegano il processo decisionale ai livelli inferiori. Quando il traffico corrisponde a una regola Pass, la valutazione ignora tutte le restanti regole di livello di amministratore per quel traffico e passa direttamente al livello. NetworkPolicy Ciò consente agli amministratori di delegare esplicitamente il controllo di determinati modelli di traffico ai team applicativi. Ad esempio, potresti utilizzare le regole Pass per delegare la gestione del traffico all'interno dello spazio dei nomi agli amministratori dello spazio dei nomi mantenendo controlli rigorosi sull'accesso esterno.

2. Livello di policy di rete: se nessun criterio di livello di amministrazione corrisponde a Nega o Consenti, o se corrisponde un'azione Pass, le risorse con ambito di namespace vengono valutate successivamente. NetworkPolicy Queste policy forniscono un controllo dettagliato all'interno dei singoli namespace e sono gestite dai team applicativi. Namespace-scoped le policy possono essere solo più restrittive delle policy di amministrazione. Non possono sovrascrivere la decisione di rifiuto di una politica di amministrazione, ma possono limitare ulteriormente il traffico consentito o passato dai criteri di amministrazione.
3. Politiche di amministrazione di livello base: se nessuna politica amministrativa o con ambito di namespace corrisponde al traffico, viene valutato il livello Baseline. ClusterNetworkPolicies Questi forniscono posizioni di sicurezza predefinite che possono essere sostituite da policy con ambito di namespace, consentendo agli amministratori di impostare impostazioni predefinite a livello di organizzazione e offrendo ai team la flessibilità di personalizzarle in base alle esigenze. Le politiche di base vengono valutate in ordine di priorità (prima il numero con la priorità più bassa).
4. Nega predefinita (se nessuna politica corrisponde): questo comportamento di negazione per impostazione predefinita garantisce che siano consentite solo connessioni esplicitamente consentite, mantenendo un elevato livello di sicurezza.

Migrazione

- Se il cluster utilizza attualmente una soluzione di terze parti per la gestione delle policy di rete di Kubernetes, è possibile utilizzare le stesse policy con il plug-in CNI di Amazon VPC per Kubernetes. Tuttavia, è necessario rimuovere la soluzione esistente in modo che non gestisca le stesse policy.

> **Warning**
>
> Dopo aver rimosso una soluzione di policy di rete, si consiglia di sostituire tutti i nodi a cui era stata applicata la soluzione di policy di rete. Questo perché le regole del traffico potrebbero essere ignorate da un pod della soluzione se questa si interrompe all'improvviso.

Installazione

- La funzionalità di policy di rete crea e richiede una `PolicyEndpoint` Custom Resource Definition (CRD) denominata `policyendpoints.networking.k8s.aws`. Gli oggetti `PolicyEndpoint` della Custom Resource sono gestiti da Amazon EKS. È sconsigliabile modificare o eliminare queste risorse.
- Se si eseguono pod che utilizzano le credenziali IAM del ruolo di istanza o si connettono all'IMDS EC2, occorre accertarsi che non siano presenti policy di rete che bloccherebbero l'accesso all'IMDS EC2. Potrebbe essere necessario aggiungere una policy di rete per consentire l'accesso all'IMDS EC2. Per ulteriori informazioni, consulta Metadati dell'istanza e dati dell'utente nella Guida per l'utente di Amazon EC2 per le istanze Linux.

  I pod che utilizzano i ruoli IAM per gli account di servizio o EKS Pod Identity non accedono all'IMDS EC2.
- Il plug-in CNI di Amazon VPC per Kubernetes non applica le policy di rete alle interfacce di rete aggiuntive per ogni pod, ma solo all'interfaccia principale per ogni pod (`eth0`). Ciò influisce sulle seguenti architetture:
  - Pod `IPv6` con la variabile `ENABLE_V4_EGRESS` impostata su `true`. Questa variabile abilita la funzionalità di uscita `IPv4` per connettere i pod IPv6 a endpoint `IPv4` come quelli esterni al cluster. La funzionalità di uscita `IPv4` opera creando un'interfaccia di rete supplementare con un indirizzo IPv4 di loopback locale.
  - In caso di utilizzo di plugin di rete concatenati come Multus. Poiché questi plug-in aggiungono interfacce di rete a ciascun pod, le policy di rete non vengono applicate ai plug-in di rete concatenati.

## Limitare il traffico di rete dei Pod con le policy di rete Kubernetes

È possibile utilizzare le policy di rete di Kubernetes per limitare il traffico di rete da e verso i pod. Per ulteriori informazioni, consulta Network Policies nella documentazione Kubernetes.

È necessario configurare quanto segue per utilizzare questa funzionalità:

1. Impostare l'applicazione delle policy all'avvio del Pod. Si effettua nel container `aws-node` del CNI `DaemonSet` di VPC.
2. Abilitare il parametro della policy di rete per il componente aggiuntivo.
3. Configura il tuo cluster per utilizzare la policy di rete Kubernetes.

Prima di iniziare, esaminare le seguenti informazioni. Per ulteriori informazioni, consulta the section called "Considerazioni".

### Prerequisiti

Di seguito sono indicati i prerequisiti per la funzionalità:

#### Versione minima del cluster

Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base. Il cluster deve eseguire una delle versioni Kubernetes e della piattaforma elencate nella tabella seguente. Sono supportate anche tutte le versioni di Kubernetes e della piattaforma successive a quelle elencate. Puoi verificare la versione corrente di Kubernetes *`my-cluster`* sostituendo il seguente comando con il nome del cluster e quindi eseguendo il comando modificato:

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

| Versione di Kubernetes | Versione della piattaforma |
|---|---|
| `1.27.4` | `eks.5` |
| `1.26.7` | `eks.6` |

Versione minima VPC CNI

Per creare sia policy di rete Kubernetes standard che policy di rete di amministrazione, devi eseguire `1.21` la versione del plugin VPC CNI. È possibile verificare la versione attuale con il seguente comando.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Se la versione è precedente alla `1.21`, consulta la pagina the section called "Aggiornamento (componente aggiuntivo EKS)" per eseguire l'aggiornamento alla versione `1.21` o successiva.

Versione minima del kernel Linux

I nodi devono avere una versione del kernel Linux `5.10` o successiva. È possibile verificare la versione del kernel con `uname -r`. Se si utilizzano le versioni più recenti delle AMI Amazon Linux ottimizzate per Amazon EKS, Amazon Linux accelerate ottimizzate per Amazon EKS e Bottlerocket, queste dispongono già della versione del kernel richiesta.

La versione `v20231116` o successive delle AMI Amazon Linux accelerate ottimizzate per Amazon EKS dispone della versione del kernel `5.10`.

Passaggio 1: Impostare l'applicazione delle policy all'avvio del Pod

Il plug-in CNI di Amazon VPC per Kubernetes configura le policy di rete per i pod parallelamente al provisioning dei pod. Fino a quando tutte le policy non saranno configurate per il nuovo pod, i container nel nuovo si avvieranno con una policy di autorizzazione predefinita. Questa è chiamata modalità standard. Una policy di autorizzazione predefinita prevede che tutto il traffico in entrata e in uscita sia consentito da e verso i nuovi pod. Ad esempio, ai pod non verrà applicata alcuna regola del firewall (tutto il traffico è consentito) finché il nuovo pod non verrà aggiornato con le policy attive.

Con la variabile `NETWORK_POLICY_ENFORCING_MODE` impostata su `strict`, i pod che utilizzano il CNI di VPC iniziano con una policy di negazione predefinita, poi vengono configurate le policy. Questa è detta modalità rigorosa. In modalità rigorosa, è necessario disporre di una policy di rete per ogni endpoint a cui i pod devono accedere nel cluster. Questo requisito si applica ai pod CoreDNS. La policy di negazione predefinita non è configurata per i pod con rete host.

È possibile modificare la policy di rete predefinita impostando la variabile di ambiente `NETWORK_POLICY_ENFORCING_MODE` su `strict` nel container `aws-node` del CNI di VPC `DaemonSet`.

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

Passaggio 2: Abilitare il parametro della policy di rete per il componente aggiuntivo

Per impostazione predefinita, per i parametri la funzionalità di policy di rete utilizza la porta 8162 sul nodo. Inoltre, la funzione utilizza la porta per le sonde sanitarie. 8163 Se esegui un'altra applicazione sui nodi o all'interno dei pod che deve utilizzare queste porte, l'app non viene eseguita. Dalla versione VPC CNI `v1.14.1` o successiva, puoi modificare queste porte.

Per abilitare il parametro della policy di rete per il componente aggiuntivo, procedere come indicato di seguito.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
3. Scegli la scheda Add-ons.
4. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
5. Nella pagina Configura **`Amazon VPC CNI`**:
   a. Selezionare una `v1.14.0-eksbuild.3` o versione successiva nell'elenco Versione.
   b. Scegliere Impostazioni di configurazione facoltative.
   c. Inserisci la chiave JSON `"enableNetworkPolicy":` e il valore `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`.

      L'esempio seguente ha la funzionalità delle policy di rete abilitata e i parametri e le sonde dell'integrità sono impostati sui numeri di porta predefiniti:

      ```
      {
          "enableNetworkPolicy": "true",
          "nodeAgent": {
              "healthProbeBindAddr": "8163",
      ```

```
        "metricsBindAddr": "8162"
    }
}
```

Helm

Se il plug-in CNI di Amazon VPC per Kubernetes è stato installato attraverso `helm`, è possibile aggiornare la configurazione per scrivere le porte.

1. Esegui il seguente comando per modificare le porte. Imposta il numero di porta nel valore, rispettivamente, della chiave `nodeAgent.metricsBindAddr` o `nodeAgent.healthProbeBindAddr`.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/aws-vpc-
cni
```

kubectl

1. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Sostituire i numeri di porta negli argomenti del comando seguente nel `args:` del container `aws-network-policy-agent` nel manifesto del daemonset `aws-node` della CNI del VPC.

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

Passaggio 3: Configurare il cluster per l'utilizzo delle policy di rete di Kubernetes

È possibile impostarlo per un componente aggiuntivo di Amazon EKS o come un componente aggiuntivo autogestito.

Componenti aggiuntivi di Amazon EKS

Utilizzando la AWS CLI, puoi configurare il cluster per utilizzare le politiche di rete Kubernetes eseguendo il comando seguente. Sostituire `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che si sta utilizzando.

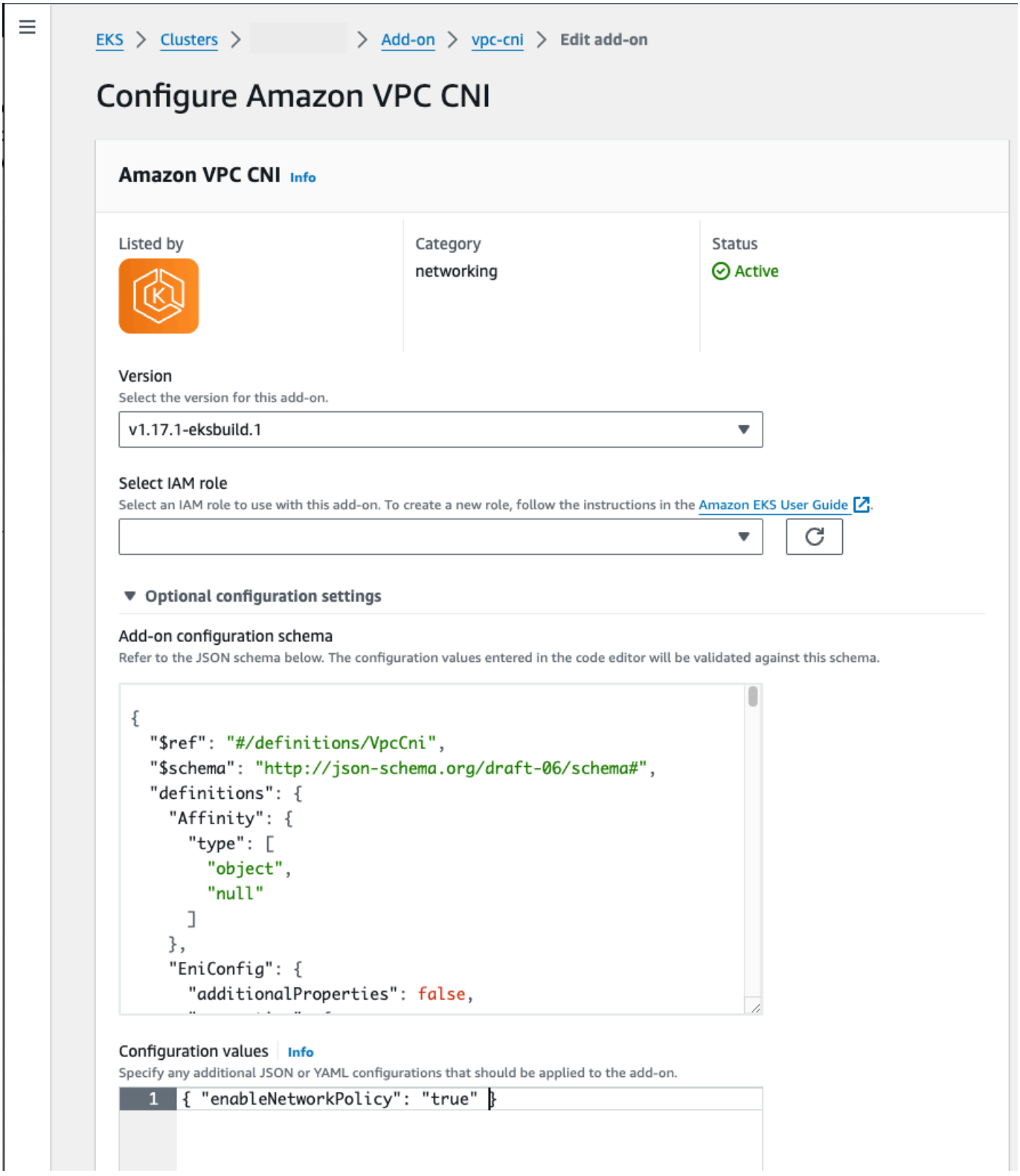
```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"enableNetworkPolicy":
 "true"}'
```

Per configurarlo utilizzando la console di AWS gestione, segui i passaggi seguenti:

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
3. Scegli la scheda Add-ons.
4. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
5. Nella pagina Configura **`Amazon VPC CNI`**:
   a. Selezionare una `v1.14.0-eksbuild.3` o versione successiva nell'elenco Versione.
   b. Scegliere Impostazioni di configurazione facoltative.
   c. Inserisci la chiave JSON `"enableNetworkPolicy":` e il valore `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`. Di seguito è riportato l'esempio dell'abilitazione della policy di rete:

      ```
      { "enableNetworkPolicy": "true" }
      ```

      Lo screenshot seguente mostra un esempio di tale scenario.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 { "enableNetworkPolicy": "true" }

Self-managed componente aggiuntivo

## Helm

Se il plug-in CNI di Amazon VPC per Kubernetes è stato installato attraverso `helm`, è possibile aggiornare la configurazione per abilitare la policy di rete.

1. Esegui il comando seguente per abilitare la policy di rete.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

## kubectl

1. Apri la `ConfigMap` del `amazon-vpc-cni` nell'editor.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

2. Aggiungi la seguente riga ai `data` nella `ConfigMap`.

```
enable-network-policy-controller: "true"
```

   Dopo aver aggiunto la riga, la `ConfigMap` dovrebbe essere simile al seguente esempio.

```
apiVersion: v1
 kind: ConfigMap
 metadata:
  name: amazon-vpc-cni
  namespace: kube-system
 data:
  enable-network-policy-controller: "true"
```

3. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

   a. Sostituire il valore `false` con `true` nell'argomento del comando `--enable-network-policy=false` in `args:` nel container `aws-network-policy-agent` nel manifesto del daemonset `aws-node` della CNI di VPC.

```
    - args:
       - --enable-network-policy=true
```

Passaggio 4. Fasi successive

Dopo aver completato la configurazione, confermare che i pod `aws-node` siano in esecuzione sul cluster.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

Di seguito viene riportato un output di esempio.

```
aws-node-gmqp7                                          2/2     Running   1 (24h ago)
 24h
aws-node-prnsh                                          2/2     Running   1 (24h ago)
 24h
```

Nei pod `aws-node` sono presenti 2 container nelle versioni `1.14` e successive. Nelle versioni precedenti e se le policy di rete sono disabilitate, nei pod `aws-node` è presente un solo container.

Ora è possibile implementare le policy di rete di Kubernetes per il cluster.

Per implementare le politiche di rete Kubernetes, puoi creare Kubernetes `NetworkPolicy` o `ClusterNetworkPolicy` oggetti e distribuirli nel tuo cluster. `NetworkPolicy`gli oggetti sono limitati a un namespace, mentre gli `ClusterNetworkPolicy` oggetti possono essere estesi all'intero cluster o a più namespace. Le policy vengono implementate per consentire o negare il traffico tra Pod in base a selettori di etichette, namespace e intervalli di indirizzi IP. Per ulteriori informazioni sulla creazione di oggetti `NetworkPolicy`, consultare Network Policies nella documentazione di Kubernetes.

L'applicazione di oggetti `NetworkPolicy` di Kubernetes viene implementata con Extended Berkeley Packet Filter (eBPF). Relativo alle implementazioni basate su `iptables`, offre caratteristiche di latenza e prestazioni inferiori, tra cui un utilizzo ridotto della CPU e l'evitamento delle ricerche sequenziali. Inoltre, le sonde eBPF forniscono l'accesso a dati contestuali che aiutano a eseguire il debug di problemi complessi a livello di kernel e a migliorare l'osservabilità. Amazon EKS supporta un BPF-based e-exporter che sfrutta le sonde per registrare i risultati delle policy su ogni nodo ed esportare i dati a raccoglitori di log esterni per facilitare il debug. Per ulteriori informazioni, consulta eBPF documentation.

Disattiva le politiche di rete Kubernetes per il traffico di rete Amazon EKS Pod

Disattiva le policy di rete Kubernetes per interrompere la limitazione del traffico di rete Amazon EKS Pod.

1. Elenca tutte le politiche di rete Kubernetes.

```
kubectl get netpol -A
```

2. Elimina ogni policy di rete Kubernetes. È necessario eliminare tutte le politiche di rete prima di disabilitarle.

```
kubectl delete netpol <policy-name>
```

3. Apri aws-node nel tuo editor DaemonSet .

```
kubectl edit daemonset -n kube-system aws-node
```

4. Sostituire il valore `true` con `false` nell'argomento del comando `--enable-network-policy` in `args:` nel container `aws-network-policy-agent` nel manifesto del daemonset `aws-node` della CNI di VPC.

```
    - args:
       - --enable-network-policy=false
```

Risoluzione dei problemi delle politiche di rete Kubernetes per Amazon EKS

Questa è la guida alla risoluzione dei problemi per la funzionalità di policy di rete di Amazon VPC CNI.

Questa guida illustra:

- informazioni su installazione, autorizzazioni di CRD e RBAC the section called "Nuove `policy, endpoint, CRD e autorizzazioni.`"
- log da esaminare durante la diagnosi dei problemi relativi alle politiche di rete the section called "Log delle policy di rete"
- esecuzione della raccolta di strumenti SDK eBPF per la risoluzione dei problemi
- Problemi noti e risoluzioni the section called "Problemi noti e risoluzioni"

> **Note**
>
> Tieni presente che le politiche di rete vengono applicate solo ai pod creati da Implementazioni di Kubernetes. Per ulteriori limitazioni delle politiche di rete nel CNI di VPC, consultare the section called "Considerazioni".

È possibile risolvere i problemi e analizzare le connessioni di rete che utilizzano le policy di rete leggendo i log della policy di rete ed eseguendo gli strumenti da eBPF SDK.

Nuove **`policy, endpoint, CRD e autorizzazioni`**.

- CRD: `policyendpoints.networking.k8s.aws`
- API di Kubernetes: `apiservice` ha chiamato `v1.networking.k8s.io`
- Risorsa di Kubernetes: `Kind: NetworkPolicy`
- RBAC: `ClusterRole` ha chiamato `aws-node` (CNI di VPC), `ClusterRole` ha chiamato `eks:network-policy-controller` (controller delle politiche di rete nel piano di controllo del cluster EKS)

Per la politica di rete, CNI di VPC crea un nuovo `CustomResourceDefinition` (CRD) chiamato `policyendpoints.networking.k8s.aws`. Il CNI VPC deve disporre delle autorizzazioni per creare il CRD e creare CustomResources (CR) di questo e dell'altro CRD installato dal VPC CNI (). `eniconfigs.crd.k8s.amazonaws.com` Entrambi i CRD sono disponibili nel file su. `crds.yaml` GitHub In particolare, CNI di VPC deve disporre delle autorizzazioni verbali "ottieni", "elenco" e "guarda" per `policyendpoints`.

La politica di rete di Kubernetes fa parte dei `apiservice` chiamato `v1.networking.k8s.io` e ciò è `apiversion: networking.k8s.io/v1` nei file YAML della politica. `DaemonSet` di CNI di VPC deve disporre delle autorizzazioni per utilizzare questa parte dell'API Kubernetes.

Le autorizzazioni di CNI di VPC sono disponibili in un `ClusterRole` chiamato `aws-node`. Si noti che gli oggetti `ClusterRole` non sono raggruppati in namespace. Di seguito, è illustrato il `aws-node` di un cluster:

```
kubectl get clusterrole aws-node -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  labels:
    app.kubernetes.io/instance: aws-vpc-cni
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: aws-node
    app.kubernetes.io/version: v1.19.4
    helm.sh/chart: aws-vpc-cni-1.19.4
    k8s-app: aws-node
  name: aws-node
rules:
- apiGroups:
  - crd.k8s.amazonaws.com
  resources:
  - eniconfigs
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
```

```
  - events.k8s.io
  resources:
  - events
  verbs:
  - create
  - patch
  - list
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
- apiGroups:
  - vpcresources.k8s.aws
  resources:
  - cninodes
  verbs:
  - get
  - list
  - watch
  - patch
```

Inoltre, un nuovo controller è avviato nel piano di controllo di ciascun cluster EKS. Il controller utilizza le autorizzazioni del `ClusterRole` chiamato `eks:network-policy-controller`. Di seguito, è illustrato il `eks:network-policy-controller` di un cluster:

```
kubectl get clusterrole eks:network-policy-controller -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: amazon-network-policy-controller-k8s
```

```
  name: eks:network-policy-controller
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/finalizers
  verbs:
  - update
```

```
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
  - patch
  - update
- apiGroups:
  - networking.k8s.io
  resources:
  - networkpolicies
  verbs:
  - get
  - list
  - patch
  - update
  - watch
```

Log delle policy di rete

Ogni decisione del CNI VPC se le connessioni sono consentite o negate da una politica di rete viene registrata nei log di flusso. I log delle policy di rete su ogni nodo includono i log di flusso per ogni pod che dispone di una policy di rete. I log delle policy di rete sono archiviati in `/var/log/aws-routed-eni/network-policy-agent.log`. Di seguito è riportato un esempio del file `network-policy-agent.log`:

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

I log della policy di rete sono disabilitati per impostazione predefinita. Per abilitare i log della policy di rete, segui i passaggi riportati di seguito:

> **Note**
>
> I log delle policy di rete richiedono 1 vCPU aggiuntiva per il container `aws-network-policy-agent` nel manifesto `aws-node DaemonSet` di CNI di VPC.

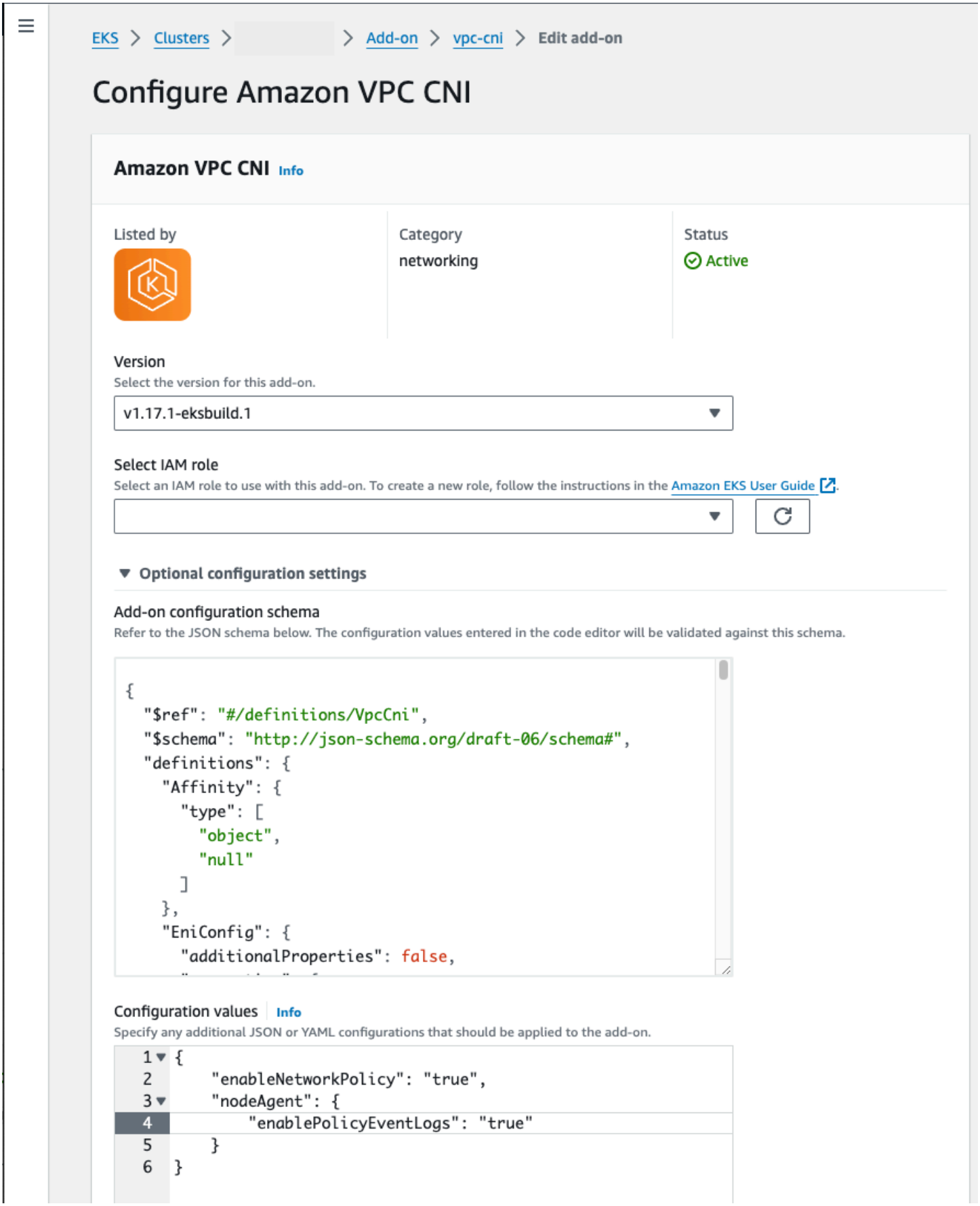
Componenti aggiuntivi di Amazon EKS

Console di gestione AWS

a. Aprire la Console Amazon EKS.
b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
c. Scegli la scheda Add-ons.
d. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
e. Nella pagina Configura ***Amazon VPC CNI***:
   i. Seleziona una `v1.14.0-eksbuild.3` o versione successiva nell'elenco a discesa Versione.
   ii. Scegli Impostazioni di configurazione facoltative.
   iii. Inserisci la chiave JSON di primo livello `"nodeAgent":` e il valore, che è un oggetto con una chiave`"enablePolicyEventLogs":` e un valore di `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. L'esempio seguente mostra i criteri di rete e i registri dei criteri di rete sono abilitati e i registri dei criteri di rete vengono inviati a Logs: CloudWatch

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

Lo screenshot seguente mostra un esempio di tale scenario.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

a. Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che stai utilizzando.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

Self-managed componente aggiuntivo

Helm

Se il plugin CNI di Amazon VPC per Kubernetes è stato installato attraverso`helm`, è possibile aggiornare la configurazione per scrivere i log della policy di rete.

a. Esegui il comando seguente per abilitare la policy di rete.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

Se il plugin CNI di Amazon VPC per Kubernetes è stato installato attraverso`kubectl`, è possibile aggiornare la configurazione per scrivere i log della policy di rete.

a. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Sostituisci il valore `false` con `true` nell'argomento del comando `--enable-policy-event-logs=false` in `args:` nel container `aws-network-policy-agent` nel manifesto `aws-node` `DaemonSet` di CNI di VPC.

```
    - args:
       - --enable-policy-event-logs=true
```

### Invia i log delle policy di rete ad Amazon CloudWatch Logs

Puoi monitorare i log delle politiche di rete utilizzando servizi come Amazon CloudWatch Logs. Puoi utilizzare i seguenti metodi per inviare i log delle politiche di rete a Logs. CloudWatch

Per i cluster EKS, i log delle policy si troveranno in `/aws/eks/`*`cluster-name`*`/cluster/`, mentre per i cluster K8S autogestiti i log verranno collocati in `/aws/k8s-cluster/cluster/`.

#### Invio dei log della policy di rete con il plugin CNI di Amazon VPC per Kubernetes

Se si abilitano i criteri di rete, viene aggiunto un secondo contenitore ai *`aws-node`* pod per un agente del nodo. Questo agente del nodo può inviare i log delle politiche di rete a CloudWatch Logs.

> **Note**
>
> Solo i log delle policy di rete vengono inviati dall'agente del nodo. Gli altri log creati dalla CNI di VPC non sono inclusi.

#### Prerequisiti

- Aggiungi le seguenti autorizzazioni come strofa o policy separata al ruolo IAM che stai utilizzando per la CNI di VPC.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

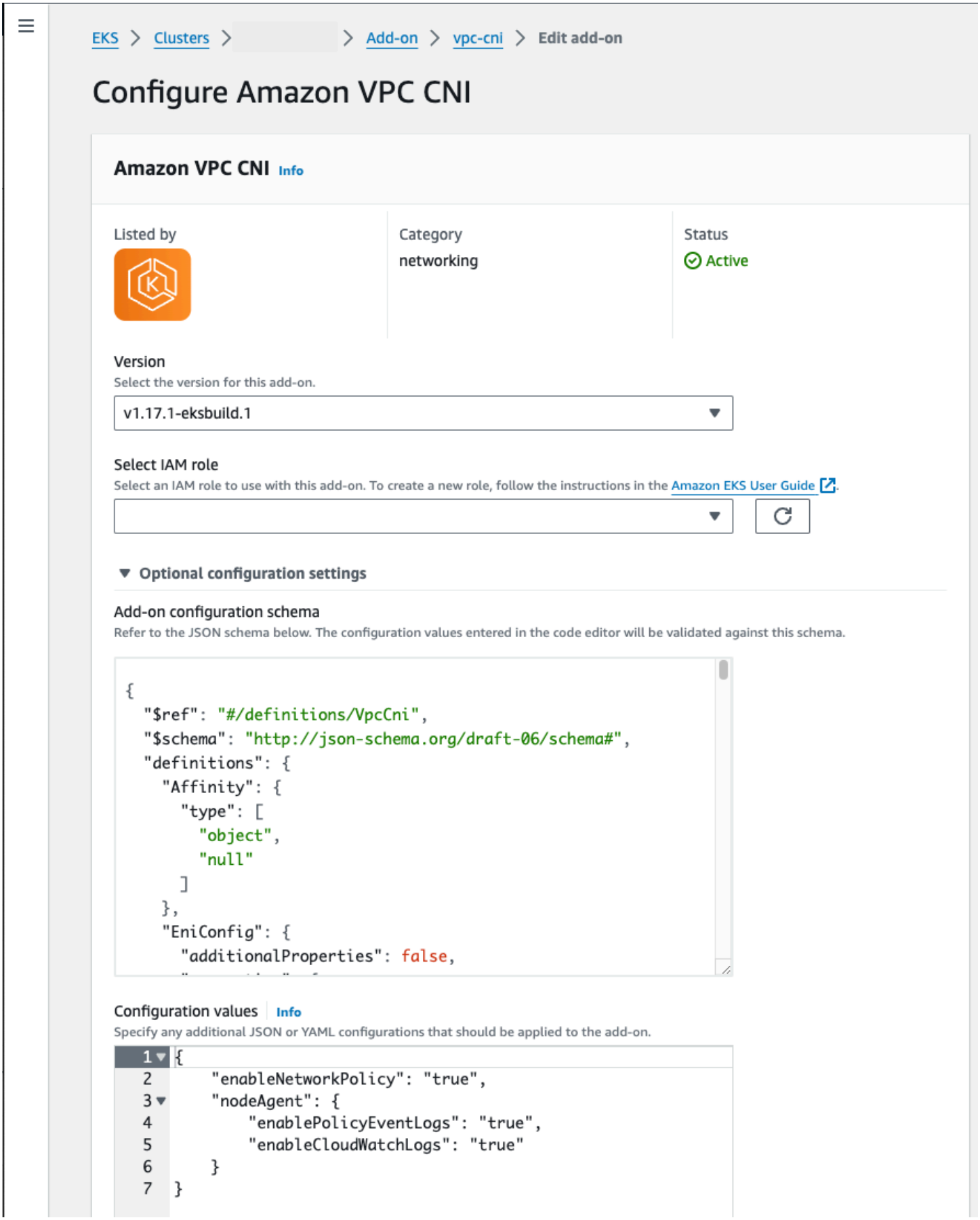
Componenti aggiuntivi di Amazon EKS

Console di gestione AWS

a. Aprire la Console Amazon EKS.
b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
c. Scegli la scheda Add-ons.
d. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
e. Nella pagina Configura ***Amazon VPC CNI***:
   i. Seleziona una `v1.14.0-eksbuild.3` o versione successiva nell'elenco a discesa Versione.
   ii. Scegli Impostazioni di configurazione facoltative.
   iii. Inserisci la chiave JSON di primo livello `"nodeAgent":` e il valore, che è un oggetto con una chiave`"enableCloudWatchLogs":` e un valore di `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. L'esempio seguente mostra i criteri di rete e i registri dei criteri di rete sono abilitati e i log vengono inviati a Logs: CloudWatch

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

Lo screenshot seguente mostra un esempio di tale scenario.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

a. Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che stai utilizzando.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

Self-managed componente aggiuntivo

timone

Se hai installato il plug-in Amazon VPC CNI per Kubernetes tramite`helm`, puoi aggiornare la configurazione per inviare i log delle politiche di rete a Logs. CloudWatch

a. Esegui il comando seguente per abilitare i log delle politiche di rete e inviarli a Logs. CloudWatch

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/aws-
vpc-cni
```

kubectl

a. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Sostituisci `false` con `true` in due argomenti del comando `--enable-policy-event-logs=false` e `--enable-cloudwatch-logs=false` nel `args:` nel container `aws-network-policy-agent` nel manifesto `aws-node` `DaemonSet` di CNI di VPC.

```
    - args:
       - --enable-policy-event-logs=true
       - --enable-cloudwatch-logs=true
```

Invio dei log delle policy di rete con Fluent Bit **`DaemonSet`**

Se stai utilizzando Fluent Bit in `DaemonSet` per inviare i log dai nodi, è possibile aggiungere una configurazione che includa i log delle policy di rete provenienti dalle policy di rete. È possibile utilizzare la configurazione di esempio seguente:

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

SDK ebPF incluso

Il plugin CNI di Amazon VPC per Kubernetes installa la raccolta di strumenti dell'SDK eBPF sui nodi. È possibile utilizzare gli strumenti dell'SDK eBPF per identificare i problemi relativi alle policy di rete. Ad esempio, il comando seguente elenca i programmi in esecuzione sul nodo.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

Per eseguire questo comando, è possibile utilizzare qualsiasi metodo di connessione al nodo.

Problemi noti e risoluzioni

Le seguenti sezioni descrivono i problemi noti con la funzionalità di policy di rete CNI di Amazon VPC e le relative soluzioni.

I log delle policy di rete sono stati generati nonostante enable-policy-event-logs sia impostato su falso

Problema: CNI di VPC EKS genera log delle politiche di rete anche quando l'impostazione `enable-policy-event-logs` è impostata su `false`.

Soluzione: l'impostazione `enable-policy-event-logs` disabilita solo i log delle "decisioni" delle politiche, ma non disabilita tutta la registrazione degli agenti della policy di rete. Questo comportamento è documentato nel file README di aws-network-policy-agent on. GitHub
Per disabilitare completamente la registrazione, potrebbe essere necessario modificare altre configurazioni di registrazione.

Problemi di pulizia della mappa della policy di rete

Problema: la rete `policyendpoint` è ancora esistente e non è ripulita dopo l'eliminazione dei pod.

Soluzione: questo problema è stato causato da un problema con il componente aggiuntivo di CNI di VPC versione 1.19.3-eksbuild.1. Per risolvere questo problema, esegui l'aggiornamento a una versione più recente del componente aggiuntivo CNI di VPC.

Le politiche di rete non sono applicate

Problema: la funzionalità della policy di rete è abilitata nel plugin CNI di Amazon VPC, ma le policy di rete non sono applicate correttamente.

Se crei una policy di rete `kind: NetworkPolicy` e questa non influisce sul pod, verifica che l'oggetto policyendpoint sia stato creato nello stesso namespace del pod. Se non ci sono oggetti `policyendpoint` nei namespace, il controller della policy di rete (parte del cluster EKS) non è stato in grado di creare regole di policy di rete per l'agente delle policy di rete (parte del CNI di VPC) da applicare.

Soluzione: la soluzione consiste nel correggere le autorizzazioni del CNI di VPC (`ClusterRole` : `aws-node`) e del controller delle policy di rete (`ClusterRole` : `eks:network-policy-controller`) e consentire queste azioni in qualsiasi strumento di applicazione delle policy come Kyverno. Assicurati che le policy di Kyverno non blocchino la creazione di oggetti `policyendpoint`. Vedi la sezione precedente per le autorizzazioni necessarie in. the section called "Nuove `policy, endpoint, CRD e autorizzazioni.`"

I pod non tornano allo stato di negazione predefinito dopo l'eliminazione della policy in modalità rigida

Problema: quando i criteri di rete sono abilitati in modalità rigida, i pod iniziano con una politica di negazione predefinita. Dopo l'applicazione delle politiche, il traffico è consentito verso gli endpoint specificati. Tuttavia, quando le policy sono eliminate, il pod non torna allo stato di negazione predefinito, invece passa allo stato di autorizzazione predefinito.

Soluzione: questo problema è stato risolto in CNI di VPC versione 1.19.3, che includeva l'agente di policy di rete versione 1.2.0. Dopo la correzione, con la modalità rigida abilitata, una volta rimosse le policy, il pod tornerà allo stato di negazione predefinito come previsto.

Latenza di avvio dei gruppi di sicurezza for pod

Problema: quando si utilizza la funzionalità dei gruppi di sicurezza per pod in EKS, la latenza di avvio dei pod aumenta.

Soluzione: la latenza è dovuta alla limitazione della velocità nel controller di risorse derivata dalla limitazione (della larghezza di banda della rete) dell'API `CreateNetworkInterface`, che il controller di risorse VPC utilizza per creare ENI di ramo per i pod. Verifica i limiti delle API del tuo account per questa operazione e, se necessario, valuta la possibilità di richiedere un aumento del limite.

FailedScheduling a causa dell'insufficiente vpc.amazonaws. com/pod-eni

Problema: i pod non riescono a pianificarsi con l'errore: `FailedScheduling 2m53s (x28 over 137m) default-scheduler 0/5 nodes are available: 5 Insufficient vpc.amazonaws.com/pod-eni. preemption: 0/5 nodes are available: 5 No preemption victims found for incoming pod.`

Soluzione: come nel problema precedente, l'assegnazione dei gruppi di sicurezza ai pod aumenta la latenza di pianificazione dei pod e può superare la soglia CNI necessaria per aggiungere ogni ENI, causando il mancato avvio dei pod. Questo è il comportamento previsto quando si utilizzano i gruppi di sicurezza per pod. Considera le implicazioni di pianificazione durante la progettazione dell'architettura del carico di lavoro.

Problemi di connettività IPAM ed errori di segmentazione

Problema: si verificano molteplici errori, tra cui problemi di connettività IPAM, richieste di limitazione ed errori di segmentazione:

- `Checking for IPAM connectivity …`
- `Throttling request took 1.047064274s`
- `Retrying waiting for IPAM-D`
- `panic: runtime error: invalid memory address or nil pointer dereference`

Soluzione: questo problema si verifica se si installa `systemd-udev` su AL2023, poiché il file è riscritto con una politica non conforme. Ciò può accadere quando si esegue l'aggiornamento a un `releasever` diverso che ha un pacchetto aggiornato o si aggiorna manualmente il pacchetto stesso. Evita l'installazione o l'aggiornamento di `systemd-udev` sui nodi AL2023.

Errore di ricerca del dispositivo in base al nome non riuscito

Problema: messaggio di errore:
`{"level":"error","ts":"2025-02-05T20:27:18.669Z","caller":"ebpf/`

```
bpf_client.go:578","msg":"failed to find device by name eni9ea69618bf0: %!
w(netlink.LinkNotFoundError={0xc000115310})"}
```

Soluzione: questo problema è stato identificato e risolto nelle versioni più recenti dell'agente di policy di rete CNI di Amazon VPC (v1.2.0). Per risolvere questo problema, effettua l'aggiornamento alla versione più recente di CNI di VPC.

Vulnerabilità di CVE nell'immagine Multus CNI

Problema: Enhanced EKS ImageScan CVE Report identifica le vulnerabilità nella versione dell'immagine Multus CNI v4.1.4-eksbuild.2_thick.

Soluzione: esegui l'aggiornamento alla nuova versione dell'immagine Multus CNI e alla nuova immagine del controller della policy di rete, che non presentano vulnerabilità. Lo scanner può essere aggiornato per risolvere le vulnerabilità rilevate nella versione precedente.

Le informazioni sul flusso NEGANO i verdetti nei log

Problema: i log delle politiche di rete mostrano i verdetti di NEGAZIONE:

```
{"level":"info","ts":"2024-11-25T13:34:24.808Z","logger":"ebpf-
client","caller":"events/events.go:193","msg":"Flow
Info: ","Src IP":"","Src Port":9096,"Dest IP":"","Dest
Port":56830,"Proto":"TCP","Verdict":"DENY"}
```

Soluzione: questo problema è stato risolto nella nuova versione di controller della policy di rete. Effettua l'aggiornamento alla versione più recente della piattaforma EKS per risolvere i problemi di registrazione.

Pod-to-pod problemi di comunicazione dopo la migrazione da Calico

Problema: dopo l'aggiornamento di un cluster EKS alla versione 1.30 e il passaggio da Calico al CNI di Amazon VPC per le policy di rete, la comunicazione da pod a pod ha esito negativo quando sono applicate le policy di rete. La comunicazione è ripristinata quando le policy di rete sono eliminate.

Soluzione: l'agente di policy di rete nel CNI di VPC non può avere tante porte specificate quante ne ha Calico. Invece, utilizza gli intervalli di porte nelle policy di rete. Il numero massimo di combinazioni uniche di porte per ogni protocollo `ingress:` o `egress:` selettore in una politica di rete è 24. Utilizza gli intervalli di porte per ridurre il numero di porte uniche ed evitare questa limitazione.

L'agente della policy di rete non supporta i pod autonomi

Problema: le policy di rete applicate ai pod autonomi possono avere un comportamento incoerente.

Soluzione: l'agente Network Policy attualmente supporta solo i pod distribuiti come parte di un. deployment/replicaset Se le policy di rete sono applicate ai pod autonomi, potrebbero esserci delle incongruenze nel comportamento. Questo è documentato nella parte superiore di questa pagina, nel e nel numero #327 di the section called "Considerazioni" GitHub aws-network-policy-agent in poi. GitHub Implementa i pod come parte di un'implementazione o di un set di replica per un comportamento coerente delle policy di rete.

Demo delle policy di rete con Stars per Amazon EKS

La demo consente di creare un servizio di front-end, di back-end e client sul cluster Amazon EKS. Inoltre, la demo consente di creare un'interfaccia utente grafica di gestione che mostra i percorsi di ingresso e uscita disponibili tra ogni servizio. Ti consigliamo di completare la demo su un cluster su cui non esegui carichi di lavoro di produzione.

Prima di creare eventuali policy di rete, tutti i servizi sono in grado di comunicare in modo bidirezionale. Dopo l'applicazione delle policy di rete, puoi vedere che il client può comunicare solo con il servizio di front-end, mentre il back-end accetta traffico solo dal front-end.

1. Applica i servizi di front-end, back-end, client e dell'interfaccia utente di gestione:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
```

2. Visualizza tutti i pod nel cluster.

```
kubectl get pods -A
```

   Di seguito viene riportato un output di esempio.

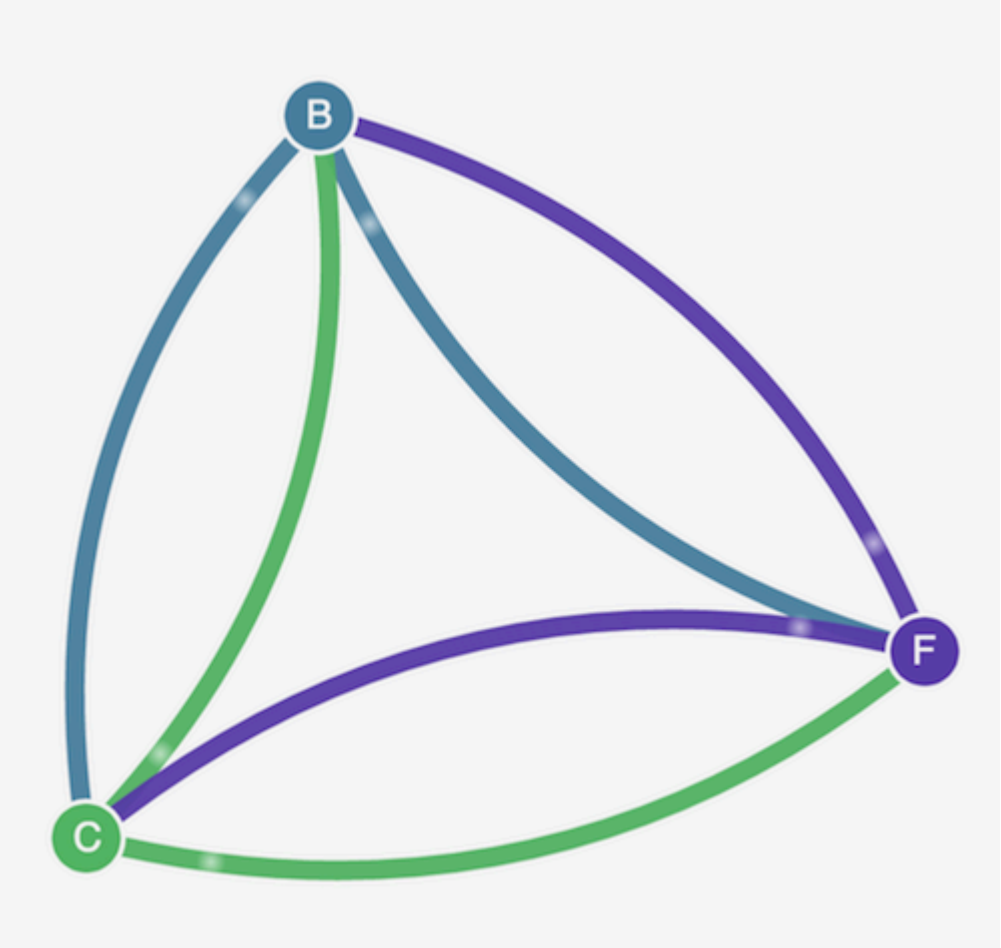
Nell'output, dovresti vedere i pod nei namespace mostrati nel seguente output. I *NAMES* tuoi pod e il numero di pod nella READY colonna sono diversi da quelli dell'output seguente. Non continuare finché non consulta pod con nomi simili e tutti hanno il valore Running nella colonna STATUS.

```
NAMESPACE         NAME                                  READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                          1/1     Running   0
     5m19s
[...]
management-ui     management-ui-qrb2g                   1/1     Running   0
     5m24s
stars             backend-sz87q                         1/1     Running   0
     5m23s
stars             frontend-cscnf                        1/1     Running   0
     5m21s
[...]
```

3. Per connetterti all'interfaccia utente di gestione, connettiti all'EXTERNAL-IP del servizio in esecuzione sul cluster:

```
kubectl get service/management-ui -n management-ui
```

4. Apri un browser nella posizione del passaggio precedente. Dovresti visualizzare l'interfaccia utente di gestione seguente. Il nodo C è il servizio client, il nodo F è il servizio front-end e il nodo B è il servizio di back-end. Ogni nodo dispone di accesso di comunicazione completo a tutti gli altri nodi, come indicato dalle righe colorate in grassetto.




5. Applica la seguente policy di rete nei due namespace `stars` e `client` per isolare tra loro i servizi:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

È possibile utilizzare i seguenti comandi per applicare la policy a entrambi i namespace:

```
kubectl apply -n stars -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

```
kubectl apply -n client -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. Aggiorna il tuo browser. Poiché l'interfaccia utente di gestione non può più raggiungere i nodi, essi non vengono visualizzati nell'interfaccia utente.
7. Applica le seguenti policy di rete per consentire all'interfaccia utente di gestione di accedere ai servizi. Applica questa policy per consentire all'interfaccia utente di:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

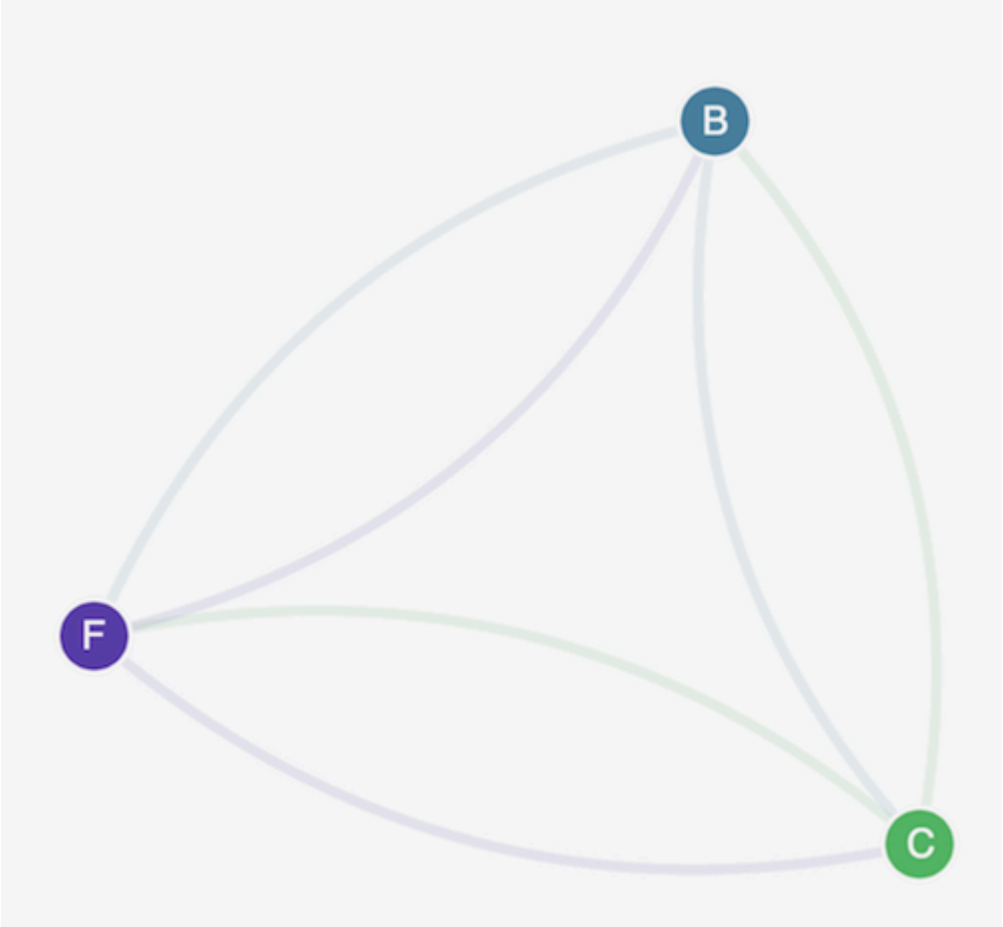
   Applica questa policy per consentire al client di:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

   È possibile utilizzare i seguenti comandi per applicare la policy a entrambe le policy:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. Aggiorna il tuo browser. L'interfaccia utente di gestione può raggiungere nuovamente i nodi, ma i nodi non sono in grado di comunicare tra loro.

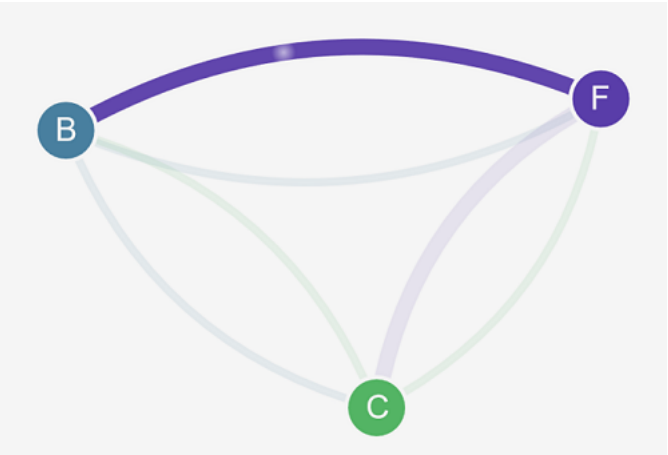



9. Applicare le seguenti policy di rete per consentire il traffico dal servizio front-end al servizio back-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
spec:
```

```
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

10 Aggiorna il tuo browser. Puoi vedere che il front-end può comunicare con il back-end.

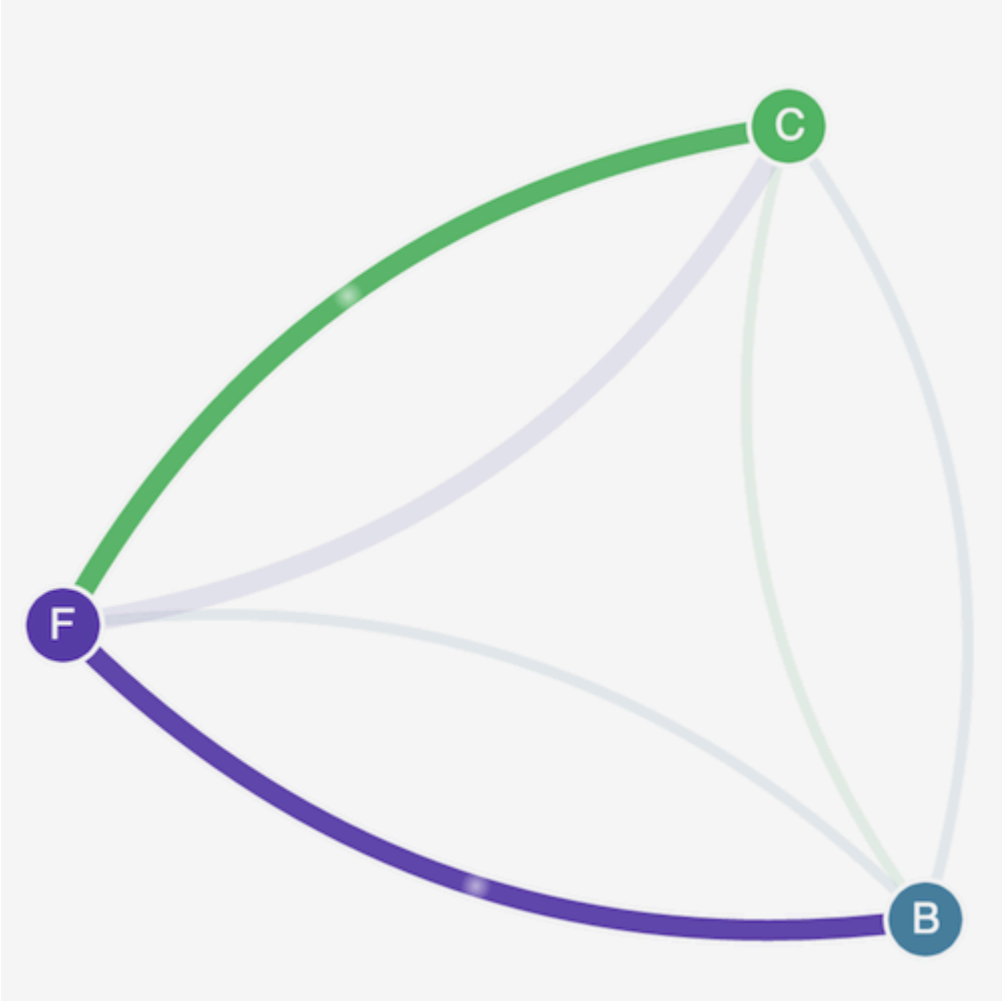



11 Applica la seguente policy di rete per consentire il traffico dal client al servizio di front-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
        - protocol: TCP
```

```
        port: 80
```

12. Aggiorna il tuo browser. Puoi vedere che il client può comunicare con il servizio di front-end. Il servizio di front-end può ancora comunicare con il servizio di back-end.




13. (Facoltativo) Al termine della demo, puoi eliminare le relative risorse.

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
```

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
```

Anche dopo l'eliminazione delle risorse, sui nodi possono essere presenti endpoint di policy di rete che potrebbero interferire in modo imprevisto con la rete nel cluster. L'unico modo sicuro per rimuovere queste regole è riavviare i nodi o terminare tutti i nodi e riciclarli. Per terminare tutti i nodi, impostare il conteggio desiderato del Gruppo Auto Scaling su 0, quindi eseguire il backup al numero desiderato oppure terminare semplicemente i nodi.

Configura la selezione della sottorete per gli indirizzi IP del Pod

Si applica a: nodi Linux con istanze Amazon EC2

Il plug-in Amazon VPC CNI per Kubernetes crea interfacce di rete elastiche secondarie (ENI) sui tuoi nodi e assegna gli indirizzi IP da tali ENI ai Pods. Per impostazione predefinita, il VPC CNI crea ENI secondari nella stessa sottorete dell'interfaccia di rete principale del nodo. È possibile controllare quali sottoreti il VPC CNI utilizza per gli indirizzi IP dei Pod tramite i seguenti metodi:

- Rilevamento avanzato delle sottoreti: il VPC CNI rileva e utilizza automaticamente le sottoreti etichettate nello stesso VPC e `kubernetes.io/role/cni` nella stessa zona di disponibilità. Richiede VPC CNI versione 1.18.0 o successiva. Consigliamo questo metodo per la maggior parte dei casi d'uso.
- Rete personalizzata: specifica manualmente le sottoreti e i gruppi di sicurezza per zona di disponibilità utilizzando risorse `ENIConfig` personalizzate. Per ulteriori informazioni, consulta the section called “Rete personalizzata”.

> **Note**
>
> Le reti personalizzate hanno la precedenza quando entrambe le funzionalità sono abilitate.

Rilevamento avanzato delle sottoreti

VPC CNI versione 1.18.0 e successive abilita l'individuazione avanzata delle sottoreti per impostazione predefinita (). `ENABLE_SUBNET_DISCOVERY=true` Il VPC CNI rileva automaticamente le sottoreti nello stesso VPC e nella stessa zona di disponibilità del nodo, quindi le utilizza per creare ENI secondari e allocare indirizzi IP Pod. Ciò espande lo spazio di indirizzi IP disponibile senza configurazione manuale. `ENIConfig`

Per verificare che la funzionalità sia abilitata:

```
kubectl describe ds aws-node -n kube-system | grep ENABLE_SUBNET_DISCOVERY
```

Per disabilitare questa funzionalità, imposta `ENABLE_SUBNET_DISCOVERY=false` su `aws-node` DaemonSet.

Comportamento dei tag di sottorete (**`kubernetes). io/role/cni`**)

Il `kubernetes.io/role/cni` tag controlla il modo in cui il VPC CNI tratta le sottoreti per le operazioni ENI e l'allocazione IP del Pod. Il tag ha effetti diversi a seconda che il VPC CNI stia creando un nuovo ENI o riconciliando uno esistente.

> **Important**
>
> I meccanismi di etichettatura Cluster-scoped ed esclusione sono disponibili solo a partire dalla versione `v1.22.2` VPC CNI. Prima di questa versione, tramite il tag era disponibile solo il comportamento di opt-in standard. `kubernetes.io/role/cni=1`

Valori Tag

La tabella seguente riassume il modo in cui ogni valore di tag influisce sul comportamento della sottorete:

| Valore di tag | Nuova creazione di ENI | Riconciliazione ENI esistente |
|---|---|---|
| 1 | Opt-in: Il VPC CNI crea nuovi ENI in questa sottorete. | L'ENI rimane disponibile per l'assegna zione di Pod IP. |

| Valore di tag | Nuova creazione di ENI | Riconciliazione ENI esistente |
|---|---|---|
| `0` | Escluso: il CNI VPC non crea nuovi ENI in questa sottorete. | Escluso: il CNI VPC esclude gli ENI esistenti in questa sottorete dall'allo cazione IP del Pod. Nessun nuovo IP Pod viene assegnato da questo ENI. |
| Assente (nessun tag) | Non utilizzato per i nuovi ENI: il VPC CNI non seleziona sottoreti secondari e senza il tag per la creazione di nuovi ENI. La sottorete principale (la sottorete in cui è stato lanciato il nodo) viene ancora utilizzata per la creazione di nuove ENI anche senza il tag, per la compatibilità con le versioni precedenti. | Nessuna interruzione: gli ENI esistenti nelle sottoreti senza tag rimangono disponibili per l'allocazione IP dei Pod. Il VPC CNI non rimuove o esclude questi ENI. |

Comportamento tra creazione e riconciliazione

Il VPC CNI applica intenzionalmente politiche diverse durante la creazione di nuovi ENI e la riconciliazione degli ENI esistenti:

- Creazione (errore chiuso per le sottoreti secondarie): quando il CNI VPC deve creare un nuovo ENI secondario, utilizza solo sottoreti secondarie etichettate in modo esplicito. `kubernetes.io/role/cni=1` Le sottoreti secondarie senza tag non vengono mai selezionate per la creazione di nuove ENI. Ciò garantisce che le nuove interfacce di rete vengano inserite solo nelle sottoreti approvate esplicitamente dagli amministratori.
- Riconciliazione (fail-open per sottoreti senza tag): quando il VPC CNI si avvia o riconcilia gli ENI esistenti già collegati al nodo, non esclude gli ENI perché la loro sottorete non ha il tag. `kubernetes.io/role/cni` In questo modo si evita l'interruzione dell'esecuzione dei Pod che già utilizzano gli indirizzi IP di tali ENI.

Il VPC CNI utilizza questo design intenzionalmente. L'esclusione forzata di un ENI già collegato da una sottorete priva di tag danneggerebbe i Pod che attualmente utilizzano gli indirizzi IP di quell'ENI.

> **Important**
>
> Per evitare che una sottorete serva nuovi IP Pod, inclusi quelli degli ENI esistenti, taggate la sottorete con. `kubernetes.io/role/cni=0` Un tag assente impedisce solo la creazione di nuovi ENI in quella sottorete. Non esclude gli ENI esistenti dall'allocazione.

### Gestione della sottorete principale

Il VPC CNI include sempre la sottorete principale del nodo (la sottorete in cui è stato avviato il nodo) per la creazione dell'ENI, anche senza il tag. `kubernetes.io/role/cni` Ciò mantiene la retrocompatibilità con i cluster esistenti. Il comportamento principale della sottorete:

- Incluso per la creazione di ENI indipendentemente dalla presenza del tag (a meno che non sia taggato`0`).
- Se contrassegnato con`kubernetes.io/role/cni=0`, il VPC CNI esclude la sottorete principale sia dalla creazione di una nuova ENI che dall'allocazione ENI esistente.

### Cluster-scoped filtraggio di sottoreti

Quando una sottorete è etichettata con`kubernetes.io/role/cni=1`, il VPC CNI verifica inoltre la presenza di tag specifici del cluster utilizzando il formato chiave. `cni.networking.k8s.aws/cluster/<cluster-name>` Se una sottorete ha tag di cluster in questo formato, solo il cluster il cui nome corrisponde utilizza quella sottorete. Le sottoreti con `kubernetes.io/role/cni=1` e senza tag specifici del cluster sono disponibili per tutti i cluster nel VPC.

Ad esempio, per limitare una sottorete a un cluster specifico:

```
aws ec2 create-tags --resources subnet-example \
  --tags Key=kubernetes.io/role/cni,Value=1 Key=cni.networking.k8s.aws/cluster/my-
cluster,Value=shared
```

Ciò è utile quando più cluster EKS condividono un VPC e si desidera che ogni cluster utilizzi sottoreti diverse per gli indirizzi IP dei Pod.

### Flusso di lavoro raccomandato

Per aggiungere nuove sottoreti per gli indirizzi IP Pod:

1. Crea nuove sottoreti nello stesso VPC e nella stessa zona di disponibilità dei tuoi nodi.

2. Etichetta le sottoreti con. `kubernetes.io/role/cni=1`
3. Assicurati che le sottoreti abbiano tabelle di routing e ACL di rete appropriati.
4. Verifica che il CNI VPC rilevi e inizi a utilizzare le nuove sottoreti.

Per rimuovere una sottorete dall'allocazione IP del Pod:

1. Etichetta la sottorete con. `kubernetes.io/role/cni=0`
2. Attendi che i Pod che utilizzano gli IP di quella sottorete terminino o vengano riprogrammati in modo naturale.
3. Verifica che il CNI VPC interrompa l'allocazione di nuovi IP Pod dagli ENI in quella sottorete.

> **Important**
>
> Non rimuovete il `kubernetes.io/role/cni` tag per smettere di usare una sottorete. La rimozione del tag impedisce la creazione di nuove ENI ma non esclude gli ENI esistenti dall'allocazione. Per escludere attivamente una sottorete, contrassegnatela con. `kubernetes.io/role/cni=0`

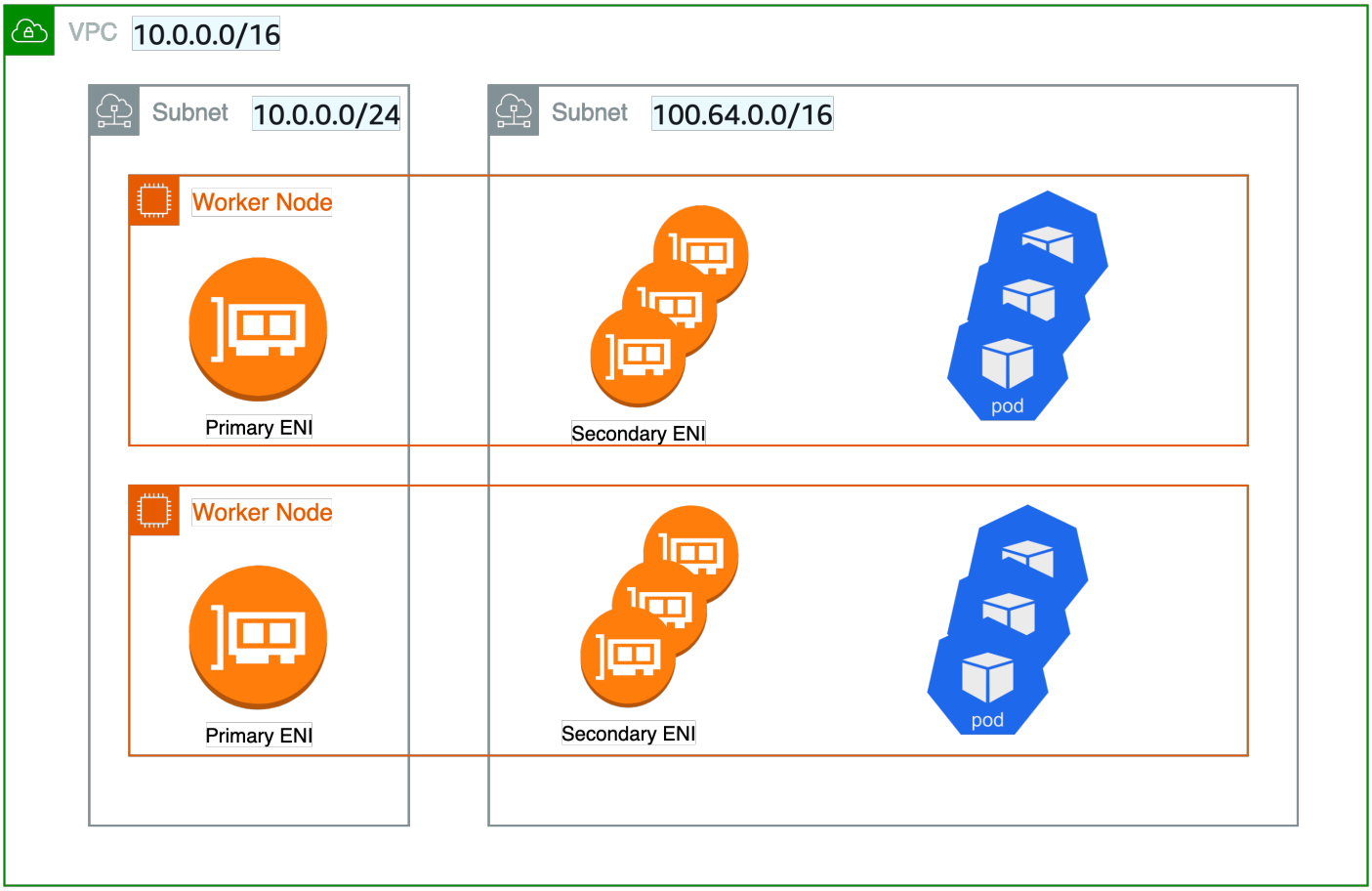
Considerazioni

- Il rilevamento avanzato delle sottoreti richiede Amazon VPC CNI versione 1.18.0 o successiva.
- La funzionalità richiede l'`ec2:DescribeSubnets`autorizzazione nel ruolo VPC CNI IAM. La policy `AmazonEKS_CNI_Policy` gestita include questa autorizzazione. La policy IAM autogestita IPv6 non la include. Se utilizzi una policy IAM autogestita (ad esempio, per i cluster IPv6), aggiungila manualmente per abilitare il rilevamento delle sottoreti. `ec2:DescribeSubnets`
- La funzionalità funziona sia con la modalità di indirizzo IP secondario che con la modalità di delega del prefisso.
- Tutte le sottoreti rilevate devono trovarsi nello stesso VPC del nodo.
- Il VPC CNI crea ENI solo nelle sottoreti che si trovano nella stessa zona di disponibilità del nodo.
- Il VPC CNI non dealloca gli ENI che hanno ancora indirizzi IP assegnati ai Pod, indipendentemente dalle modifiche ai tag.
- Quando si utilizzano VPC condivisi (sottoreti multiaccount), tagga le sottoreti nell'account del partecipante in cui viene avviato il cluster.

- È possibile utilizzare il rilevamento avanzato delle sottoreti insieme ai gruppi di sicurezza per i pod, le politiche di rete, la delega dei prefissi e SNAT.

Implementazione dei pod in sottoreti alternative con reti personalizzate

Si applica a: nodi Fargate Linux `IPv4`, nodi Linux con istanze Amazon EC2




Per impostazione predefinita, il plug-in CNI di Amazon VPC per Kubernetes crea interfacce di rete elastiche secondarie (interfacce di rete) per il nodo Amazon EC2 nella stessa sottorete dell'interfaccia di rete primaria del nodo. Inoltre, associa all'interfaccia di rete secondaria gli stessi gruppi di sicurezza associati a quella primaria. È possibile far sì che il plugin crei interfacce di rete secondarie in una sottorete diversa o associ gruppi di sicurezza diversi alle interfacce di rete secondarie, o entrambe le cose, in base a uno dei motivi riportati di seguito:

- È disponibile un numero limitato di indirizzi `IPv4` nella sottorete in cui si trova l'interfaccia di rete primaria. Ciò potrebbe limitare il numero di Pod che è possibile creare nella sottorete. Utilizzando

una sottorete diversa per le interfacce di rete secondarie, puoi aumentare il numero di indirizzi `IPv4` disponibili per i Pod.

- Per motivi di sicurezza, i tuoi Pod dovrebbero usare sottoreti o gruppi di sicurezza diversi rispetto all'interfaccia di rete primaria del nodo.
- I nodi sono configurati in sottoreti pubbliche e puoi collocare i Pod in sottoreti private. La tabella di instradamento associata a una sottorete pubblica include un percorso a un gateway Internet. La tabella di routing associata a una sottorete privata non include una route a un gateway Internet.

> **Tip**
>
> Puoi anche aggiungere una sottorete nuova o esistente direttamente al cluster Amazon EKS, senza utilizzare reti personalizzate. Per ulteriori informazioni, consulta the section called "Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione".

Considerazioni

Di seguito sono riportate alcune considerazioni sull'utilizzo della funzionalità.

- L'abilitazione di una rete personalizzata non consente di assegnare ai Pod gli indirizzi IP associati all'interfaccia di rete primaria. Solo gli indirizzi IP delle interfacce di rete secondarie vengono assegnati ai Pod.
- Se il cluster utilizza la famiglia `IPv6`, non puoi usare la rete personalizzata.
- Se si prevede di utilizzare una rete personalizzata esclusivamente per ridurre l'esaurimento dell'indirizzo `IPv4`, è invece possibile creare un cluster tramite la famiglia `IPv6`. Per ulteriori informazioni, consulta the section called "IPv6".
- I Pod implementati nelle sottoreti specificate per le interfacce di rete secondarie possono utilizzare sottoreti e gruppi di sicurezza diversi rispetto all'interfaccia di rete primaria del nodo, a patto che si trovino nello stesso VPC del nodo.
- Per Fargate, le sottoreti sono controllate tramite il profilo Fargate. Per ulteriori informazioni, consulta the section called "Definisci i profili".

Personalizzare l'interfaccia di rete secondaria nei nodi di Amazon EKS

Prima di iniziare questo tutorial, completa le seguenti operazioni:

- Esaminare le considerazioni
- Familiarità con il modo in cui il plug-in CNI di Amazon VPC per Kubernetes crea interfacce di rete secondarie e assegna indirizzi IP ai Pod. Per ulteriori informazioni, consulta Allocazione di ENI su GitHub.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Ti consigliamo di completare la procedura descritta in questo argomento in una shell Bash. In alternativa, puoi apportare alcune modifiche alla tua shell per alcuni comandi di script, come i caratteri di continuazione della riga, e per il modo in cui le variabili vengono impostate e utilizzate. Inoltre, le regole di escape e di utilizzo delle virgolette per la shell (interprete di comandi) potrebbero essere diverse. Per ulteriori informazioni, consulta Uso delle virgolette con le stringhe nella AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

Per questo tutorial, ti consigliamo di utilizzare i valori di esempio tranne quando è indicato di sostituirli. Puoi sostituire qualsiasi valore di esempio al completamento delle fasi dedicate alla creazione del cluster. Consigliamo di completare tutte le fasi nello stesso terminale, dato che le variabili impostate e utilizzate durante la procedura non sono presenti in terminali diversi.

I comandi in questo argomento sono formattati utilizzando le convenzioni elencate in Utilizzo degli esempi CLI AWS. Se esegui comandi dalla riga di comando su risorse che si trovano in una AWS regione diversa da quella predefinita definita nel profilo AWS CLI che stai utilizzando, devi aggiungerli `--region us-west-2` ai comandi, sostituendoli `us-west-2` con la tua AWS AWS regione.

Per implementare le reti personalizzate nel cluster di produzione, passa alla the section called "Passaggio 2: configurazione del VPC".

Passaggio 1: creazione di un VPC e di un cluster di test

Le procedure seguenti illustrano la creazione di un VPC e un cluster di test, oltre alla configurazione di una rete personalizzata per tale cluster. Consigliamo di non utilizzare il cluster di test per i carichi di lavoro di produzione poiché alcune funzionalità da utilizzare con il cluster di produzione non sono trattate in questo argomento. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".

1. Eseguire il seguente comando per definire la variabile `account_id`.

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. Crea un VPC.

   a. Se esegui la distribuzione su un sistema di test, crea un VPC utilizzando un modello Amazon AWS CloudFormation EKS.

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

   b. La AWS CloudFormation creazione dello stack richiede alcuni minuti. Utilizzare il comando seguente per verificare lo stato di implementazione dello stack.

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

   Non passare alla Passaggio successiva finché l'output del comando non è `CREATE_COMPLETE`.

   c. Definisci le variabili con i valori degli ID delle sottoreti private creati dal modello.

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
```

```
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

d. Definire le variabili con le zone di disponibilità delle sottoreti recuperate nel passaggio precedente.

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
 'Subnets[*].AvailabilityZone' --output text)
```

3. Crea un ruolo IAM del cluster.

a. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crea il ruolo IAM del cluster Amazon EKS. Se necessario, inserisci il prefisso `eks-cluster-role-trust-policy.json` nel percorso del computer in cui hai eseguito la scrittura del file nella Passaggio precedente. Il comando associa il criterio di attendibilità creato nella Passaggio precedente al ruolo. Per creare un ruolo IAM, è necessario assegnare l'azione `iam:CreateRole` (autorizzazione) al principale IAM che sta creando il ruolo.

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --assume-
role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. Allega al ruolo la policy gestita da Amazon EKS, denominata AmazonEKSClusterPolicy. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Crea un cluster Amazon EKS e configura il dispositivo per comunicare con esso.

   a. Crea un cluster.

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws: iam::$account_id:role/
myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds="$subnet_id_1","$subnet_id_2"
```

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

   b. La creazione del cluster richiede diversi minuti. Utilizzare il comando seguente per verificare lo stato di implementazione del cluster.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

   Non passare alla Passaggio successiva finché l'output del comando non è `"ACTIVE"`.

   c. Configura `kubectl` per comunicare con il cluster.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

Passaggio 2: configurazione del VPC

Questo tutorial richiede il VPC creato nella the section called "Passaggio 1: creazione di un VPC e di un cluster di test". Per un cluster di produzione, modificare i passaggi in base al VPC, sostituendo tutti i valori di esempio con i propri.

1. Accertati che la versione del plug-in CNI di Amazon VPC per Kubernetes installata sia l'ultima versione. Per determinare la versione più recente per il componente aggiuntivo del tipo Amazon EKS e aggiornarla, consulta la pagina the section called "Aggiornamento di un componente aggiuntivo". Per determinare la versione più recente per il componente aggiuntivo del tipo autogestito e aggiornarla, consulta la pagina the section called "CNI di Amazon VPC".
2. Recupera l'ID del VPC in cui si trova il cluster e memorizzarlo in una variabile per l'utilizzo nei passaggi successivi.

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. Associa un blocco CIDR (Classless Inter-Domain Routing) aggiuntivo al VPC del cluster. L'intervallo CIDR non può sovrapporsi agli intervalli CIDR esistenti già associati.

   a. Visualizza i blocchi CIDR attualmente associati al tuo VPC.

   ```
   aws ec2 describe-vpcs --vpc-ids $vpc_id \
       --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
    CidrBlockState.State}' --out table
   ```

   Di seguito viene riportato un output di esempio:

   ```
   ---------------------------------
   |          DescribeVpcs          |
   +-----------------+--------------+
   |    CIDRBlock    |    State     |
   +-----------------+--------------+
   |  192.168.0.0/24 |  associated  |
   +-----------------+--------------+
   ```

   b. Associa un blocco CIDR aggiuntivo al VPC. Sostituire il valore dell'intervallo CIDR nel comando seguente. Per ulteriori informazioni, consulta Associate additional IPv4 CIDR blocks with your VPC nella Guida per l'utente di Amazon VPC.

   ```
   aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
   ```

   c. Verifica che il nuovo blocco sia associato.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

Di seguito viene riportato un output di esempio.

```
---------------------------------
|          DescribeVpcs         |
+-----------------+-------------+
|    CIDRBlock    |    State    |
+-----------------+-------------+
|  192.168.0.0/24 |  associated |
|  192.168.1.0/24 |  associated |
+-----------------+-------------+
```

Non andare al passaggio successivo finché il nuovo `State` dell'intervallo CIDR non è `associated`.

4. Crea tutte le sottoreti che vuoi utilizzare in ogni zona di disponibilità in cui si trovano le sottoreti esistenti. Specificare un intervallo CIDR all'interno dell'intervallo CIDR associato al VPC in una Passaggio precedente.

   a. Crea nuove sottoreti. Sostituire i valori dell'intervallo CIDR nel seguente comando. È necessario creare le sottoreti in un blocco CIDR VPC diverso rispetto alle sottoreti esistenti, ma all'interno delle stesse zone di disponibilità. In questo esempio si crea una sottorete nel nuovo blocco CIDR in ogni zona di disponibilità in cui esistono le sottoreti private attuali. Gli ID delle sottoreti create vengono memorizzati in variabili da utilizzare nei passaggi successivi. I valori `Name` corrispondono a quelli assegnati alle sottoreti create utilizzando il modello Amazon EKS VPC in un passaggio precedente. I nomi non sono obbligatori. È possibile utilizzare nomi diversi.

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_1
 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_2
 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> Per impostazione predefinita, le nuove sottoreti sono implicitamente associate alla tabella di routing principale del VPC. Questa tabella di instradamento consente la comunicazione tra tutte le risorse implementate nel VPC, ad eccezione delle risorse con indirizzi IP esterni ai blocchi CIDR associate al VPC. Per modificare questo comportamento, associa la tabella di instradamento alle sottoreti. Per ulteriori informazioni, consulta Tabelle di instradamento per sottoreti nella Guida per l'utente di Amazon VPC.

b. Per visualizzare le sottoreti attuali nel tuo VPC.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

Di seguito viene riportato un output di esempio:

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+---------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId          |
+------------------+--------------------+---------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1       |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2       |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3       |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4       |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5       |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6       |
+------------------+--------------------+---------------------------+
```

Come puoi vedere, le sottoreti create nel blocco CIDR `192.168.1.0` si trovano nelle stesse zone di disponibilità di quelle create nel blocco CIDR `192.168.0.0`.

Passaggio 3: configurazione delle risorse Kubernetes

1. Imposta la variabile di `AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG` ambiente su in. `true` `aws-node` DaemonSet

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. Recupera l'ID del gruppo di sicurezza del cluster e memorizzalo in una variabile da utilizzare nel passaggio successivo. Amazon EKS crea automaticamente questo gruppo di sicurezza durante la creazione del cluster.

```
cluster_security_group_id=$(aws eks describe-cluster --name my-custom-networking-
cluster --query cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. Creare una risorsa personalizzata `ENIConfig` per ogni sottorete in cui vuoi implementare i Pod.

   a. Crea un file univoco per ogni configurazione di interfaccia di rete.

      I comandi seguenti creano file `ENIConfig` separati per le due sottoreti create in una Passaggio precedente. Il valore di `name` deve essere univoco. Il nome corrisponde alla zona di disponibilità in cui si trova la sottorete. Il gruppo di sicurezza del cluster viene assegnato a `ENIConfig`.

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
```

```
  subnet: $new_subnet_id_2
EOF
```

Per un cluster di produzione, puoi apportare le seguenti modifiche ai comandi precedenti:

- Sostituire $cluster_security_group_id con l'ID di un gruppo di sicurezza esistente da utilizzare per ciascun `ENIConfig`.
- Quando possibile, consigliamo di assegnare a `ENIConfigs` lo stesso nome della zona di disponibilità che verrà utilizzato per `ENIConfig`. Per una serie di motivi, tuttavia, potrebbe essere necessario utilizzare nomi diversi per `ENIConfigs` e le zone di disponibilità. Ad esempio, se nella stessa zona di disponibilità sono presenti più di due sottoreti e si desidera utilizzarle entrambe con reti personalizzate, saranno necessari più `ENIConfigs` per la stessa zona di disponibilità. Dal momento che ogni `ENIConfig` richiede un nome univoco, non puoi assegnare lo stesso nome della zona di disponibilità a più `ENIConfigs`.

  Se i nomi di `ENIConfig` non sono tutti uguali a quelli delle zone di disponibilità, sostituire $az_1 e $az_2 con i nomi scelti nei comandi precedenti e annotare i nodi con ENIConfig come descritto più avanti in questo tutorial.

  > **Note**
  >
  > Nel caso in cui non specifichi un gruppo di sicurezza valido da utilizzare con un cluster di produzione:

- se utilizzi la versione `1.8.0` o successiva del plug-in CNI di Amazon VPC per Kubernetes, vengono impiegati i gruppi di sicurezza associati all'interfaccia di rete elastica primaria del nodo.
- se utilizzi una versione del plug-in CNI di Amazon VPC per Kubernetes precedente alla `1.8.0`, il gruppo di sicurezza di default per il VPC viene assegnato alle interfacce di rete secondarie.

> **Important**
>
> - `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` è un'impostazione predefinita per la configurazione del plugin CNI di Amazon VPC per Kubernetes. Per impostazione predefinita, il traffico destinato agli indirizzi IP che non si trovano all'interno di uno dei blocchi CIDR associati al VPC utilizza i gruppi di sicurezza e le sottoreti dell'interfaccia di rete primaria del nodo. Le sottoreti e i gruppi di sicurezza definiti in

`ENIConfigs` al fine di creare interfacce di rete secondarie non vengono utilizzati per questo traffico. Per ulteriori informazioni su questa impostazione, consulta the section called "Traffico in uscita".

- Se utilizzi gruppi di sicurezza anche per i Pod, il gruppo di sicurezza specificato in una `SecurityGroupPolicy` viene utilizzato al posto di quello specificato in `ENIConfigs`. Per ulteriori informazioni, consulta the section called "Gruppi di sicurezza per pod".

b. Applica ogni file delle risorse personalizzato creato in precedenza al cluster con i comandi seguenti.

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. Verifica che `ENIConfigs` sia stato creato.

```
kubectl get ENIConfigs
```

Di seguito viene riportato un output di esempio.

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. Se stai abilitando la rete personalizzata su un cluster di produzione ed è stato assegnato a `ENIConfigs` un nome diverso da quello dalla zona di disponibilità per cui lo si sta utilizzando, passa alla Passaggio successiva per implementare i nodi Amazon EC2.

Consenti a Kubernetes di applicare automaticamente `ENIConfig` per una zona di disponibilità a tutti i nuovi nodi Amazon EC2 creati nel cluster.

a. Per il cluster di test in questo tutorial, passa alla Passaggio successiva.

Per un cluster di produzione, controlla se esiste un'annotazione con la chiave `k8s.amazonaws.com/eniConfig` per la variabile di `ENI_CONFIG_ANNOTATION_DEF` ambiente nelle specifiche del contenitore per. `aws-node` DaemonSet

```
kubectl describe daemonset aws-node -n kube-system | grep
 ENI_CONFIG_ANNOTATION_DEF
```

Se viene restituito un output, l'annotazione è presente. Se non viene restituito alcun output, la variabile non viene impostata. Per un cluster di produzione, puoi scegliere di utilizzare la presente impostazione o quella riportata nel passaggio successivo. L'utilizzo di questa impostazione sovrascrive quella del passaggio successivo. In questo tutorial viene utilizzata l'impostazione riportata nel passaggio successivo.

b. Aggiorna il tuo `aws-node` DaemonSet `ENIConfig` per applicare automaticamente la zona di disponibilità a tutti i nuovi nodi Amazon EC2 creati nel tuo cluster.

```
kubectl set env daemonset aws-node -n kube-system
 ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
```

Passaggio 4: implementazione di nodi Amazon EC2

1. Creare un ruolo IAM del nodo.

   a. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   b. Creare un ruolo IAM e memorizzare il nome della risorsa Amazon (ARN) restituito in una variabile da utilizzare in una Passaggio successiva.

```
node_role_arn=$(aws iam create-role --role-name myCustomNetworkingNodeRole --
assume-role-policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

   c. Allega al ruolo IAM le tre policy gestite IAM richieste.

```
aws iam attach-role-policy \
```

```
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name myCustomNetworkingNodeRole
```

> **Important**
>
> Per semplicità, in questo tutorial la policy AmazonEKS_CNI_Policy è allegata al ruolo IAM del nodo. Tuttavia, in un cluster di produzione, consigliamo di allegare la policy a un ruolo IAM separato da utilizzare solo con il plug-in CNI di Amazon VPC per Kubernetes. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

2. Creare uno dei seguenti tipi di gruppi di nodi. Per determinare il tipo di istanza da implementare, consulta the section called "Tipi di istanza Amazon EC2". Per questo tutorial, utilizza l'opzione Gestito, senza un modello di avvio o con un modello di avvio senza un ID AMI specificato. Se prevedi di utilizzare il gruppo di nodi per i carichi di lavoro di produzione, ti consigliamo di familiarizzare con tutte le opzioni del gruppo di nodi gestito e del gruppo di nodi autogestito prima di implementare il gruppo di nodi.
   - Gestito: implementare il gruppo di nodi utilizzando una delle opzioni seguenti:
     - Senza un modello di avvio o con un modello di avvio senza un ID AMI specificato: esegui il comando seguente. Per questo tutorial, utilizza i valori di esempio. Per un gruppo di nodi di produzione, sostituire tutti i valori di esempio con i propri. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti.

       ```
       aws eks create-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
       name my-nodegroup \
           --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
        $node_role_arn
       ```

     - Con un modello di avvio con un ID AMI specificato

       A. Determina il numero massimo di Pod per i tuoi nodi in base al tipo di istanza. Per ulteriori informazioni, consulta the section called "Come viene determinato MaxPods". Nota il valore da utilizzare nel passaggio successivo.

B. Nel modello di avvio, specifica un ID AMI ottimizzato per Amazon EKS o un'AMI personalizzata sviluppata con l'AMI ottimizzato per Amazon EKS, quindi implementa il gruppo di nodi utilizzando un modello di avvio e fornisci i seguenti dati utente nel modello di avvio. Questi dati utente passano gli argomenti alla `NodeConfig` specifica. Per ulteriori informazioni in merito NodeConfig, consulta il riferimento all'NodeConfig API. Puoi sostituire `20` con il valore del passaggio precedente (consigliato) o con un valore desiderato.

```
---
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"
--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    ...
    kubelet:
      config:
        maxPods: 20
```

Se hai creato un'AMI personalizzata che non è stata sviluppata con quella ottimizzata per Amazon EKS, devi creare autonomamente la configurazione.

- Self-managed

  i. Determina il numero massimo di Pod per i tuoi nodi in base al tipo di istanza. Per ulteriori informazioni, consulta the section called "Come viene determinato MaxPods". Nota il valore da utilizzare nel passaggio successivo.

  ii. Implementa il gruppo di nodi utilizzando le istruzioni in the section called "Amazon Linux".

> **Note**
>
> Se desideri che i nodi di un cluster di produzione supportino un numero significativamente maggiore di Pod, puoi abilitare la delega dei prefissi. Ad esempio, per un tipo di istanza `m5.large` viene restituito `110`. Per istruzioni su come abilitare questa funzionalità,

consulta the section called "Aumentare gli indirizzi IP". Puoi utilizzare questa funzionalità con una rete personalizzata.

3. La creazione di gruppi di nodi richiede diversi minuti. Puoi verificare lo stato della creazione di un gruppo di nodi gestiti con il comando seguente.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

   Non andare al passaggio successivo finché l'output restituito non è `ACTIVE`.

4. Ai fini di questo tutorial, puoi ignorare questo passaggio.

   Per un cluster di produzione, se è stato assegnato a `ENIConfigs` un nome diverso da quello della zona di disponibilità in cui lo stai utilizzando, devi annotare i nodi con il nome `ENIConfig` da utilizzare con il nodo. Questo passaggio non è necessario se disponi di una sola sottorete in ogni zona di disponibilità e hai assegnato a `ENIConfigs` gli stessi nomi delle zone di disponibilità. Questo perché il plug-in CNI di Amazon VPC per Kubernetes associa automaticamente il `ENIConfig` corretto al nodo se questa opzione è stata abilitata in una Passaggio precedente.

   a. Ottieni l'elenco dei nodi nel cluster.

```
kubectl get nodes
```

      Di seguito viene riportato un output di esempio:

```
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal    Ready    <none>   8m49s   v1.22.9-
eks-810597c
ip-192-168-0-92.us-west-2.compute.internal     Ready    <none>   8m34s   v1.22.9-
eks-810597c
```

   b. Determina in quale zona di disponibilità si trova ogni nodo. Eseguire il comando seguente per ogni nodo restituito nella Passaggio precedente, sostituendo gli indirizzi IP in base all'output precedente.

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-
name,Values=ip-192-168-0-126.us-west-2.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone: Placement.AvailabilityZone,
 SubnetId: SubnetId}'
```

Di seguito viene riportato un output di esempio.

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. Annota ogni nodo con la risorsa `ENIConfig` creata per l'ID della sottorete e la zona di disponibilità. Puoi annotare solo un nodo con un `ENIConfig`, ma più nodi con lo stesso `ENIConfig`. Sostituire i valori di esempio con i propri valori.

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

5. Se in un cluster di produzione erano presenti nodi con Pod in esecuzione prima di passare all'utilizzo della funzionalità di rete personalizzata, completare le attività seguenti:
   a. Assicurati di avere dei nodi disponibili abilitati per la funzionalità di rete personalizzata.
   b. Cordonare ed espellere i nodi per interrompere normalmente i Pod. Per ulteriori informazioni, consulta Safely Drain a Node (Espellere un nodo in modo sicuro) nella documentazione Kubernetes.
   c. Termina i nodi. Se i nodi si trovano in un gruppo di nodi gestito esistente, puoi eliminare il gruppo di nodi. Eseguire il seguente comando seguente.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup
```

   Solo i nuovi nodi registrati con l'etichetta `k8s.amazonaws.com/eniConfig` useranno la nuova funzionalità di rete personalizzata.
6. Verificare che ai Pod venga assegnato un indirizzo IP da un intervallo CIDR associato a una delle sottoreti create in una Passaggio precedente.

```
kubectl get pods -A -o wide
```

   Di seguito viene riportato un output di esempio.

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
           NODE                                                NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4                1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp                1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr      1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9      1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq              1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk              1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

Puoi vedere che ai coredns vengono assegnati indirizzi IP dall'intervallo CIDR `192.168.1.0` aggiunto al VPC. Senza una rete personalizzata, ai pod sarebbero stati assegnati indirizzi provenienti dall'unico blocco CIDR originariamente associato al VPC, ossia il blocco CIDR `192.168.0.0`.

Se una `spec` del Pod contiene il valore `hostNetwork=true`, viene assegnato l'indirizzo IP principale del nodo, non un indirizzo proveniente dalle sottoreti aggiunte. Per impostazione predefinita, questo valore è impostato su `false`. Questo valore è impostato su `true` per il `kube-proxy` e i pod (`aws-node`) del plug-in CNI di Amazon VPC per Kubernetes in esecuzione sul cluster. Per questo motivo al `kube-proxy` e ai pod `aws-node` del plug-in non vengono assegnati indirizzi 192.168.1.x nell'output precedente. Per ulteriori informazioni sull'`hostNetwork`impostazione di un Pod, consulta PodSpec v1 core nel riferimento all'API Kubernetes.

Passaggio 5: eliminazione delle risorse del tutorial

Ti consigliamo di eliminare le risorse create dopo aver completato il tutorial. In seguito, potrai modificare i passaggi per abilitare la rete personalizzata di un cluster di produzione.

1. Se il gruppo di nodi creato era solo a scopo di test, eliminalo.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name
 my-nodegroup
```

2. Anche dopo che l'output della AWS CLI indica che il cluster è stato eliminato, il processo di eliminazione potrebbe non essere effettivamente completo. Questo processo di eliminazione richiede alcuni minuti. Verificare che il processo sia completo con il comando seguente.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

   Non continuare finché l'output restituito non è simile al seguente.

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

3. Se il gruppo di nodi creato era solo a scopo di test, elimina il ruolo IAM del nodo.

   a. Scollega le policy dal ruolo.

```
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
```

   b. Elimina il ruolo.

```
aws iam delete-role --role-name myCustomNetworkingNodeRole
```

4. Elimina il cluster.

```
aws eks delete-cluster --name my-custom-networking-cluster
```

Verifica l'eliminazione del cluster con il comando seguente.

```
aws eks describe-cluster --name my-custom-networking-cluster --query cluster.status
 --output text
```

Quando viene restituito un output simile al seguente, il cluster è stato eliminato correttamente.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-custom-networking-cluster.
```

5. Elimina il ruolo IAM del cluster.
   a. Scollega le policy dal ruolo.

   ```
   aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
   policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
   ```

   b. Elimina il ruolo.

   ```
   aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
   ```

6. Elimina le sottoreti create in una Passaggio precedente.

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

7. Elimina il VPC creato.

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

## Assegnazione di più indirizzi IP ai nodi Amazon EKS con prefissi

Si applica a: nodi Linux e Windows con istanze Amazon EC2

Si applica a: sottoreti pubbliche e private

Ogni istanza Amazon EC2 supporta un numero massimo di interfacce di rete elastiche e un numero massimo di indirizzi IP che possono essere assegnati a ogni interfaccia di rete. Ogni nodo richiede un indirizzo IP per ogni interfaccia di rete. Tutti gli altri indirizzi IP disponibili possono essere assegnati a `Pods`. Ogni `Pod` necessita di un proprio indirizzo IP. Di conseguenza, si potrebbero avere nodi con

risorse di calcolo e memoria disponibili, ma che non possono ospitare ulteriori `Pods` perché il nodo ha esaurito gli indirizzi IP a cui assegnare `Pods`.

È possibile aumentare sensibilmente il numero di indirizzi IP che i nodi possono assegnare ai `Pods` assegnando prefissi IP ai nodi, anziché assegnare singoli indirizzi IP secondari. Ogni prefisso include diversi indirizzi IP. Se non configuri il cluster per l'assegnazione del prefisso IP, il cluster deve effettuare più chiamate di interfaccia di programmazione dell'applicazione (API) di Amazon EC2 per la configurazione delle interfacce di rete e degli indirizzi IP necessari alla connettività del Pod. La frequenza di queste chiamate API, man mano che i cluster acquisiscono dimensioni maggiori, può portare a tempi di avvio di Pod e delle istanze più lunghi. Ciò si traduce in ritardi di dimensionamento per soddisfare la domanda di carichi di lavoro grandi e spigolosi e aggiunge costi generali di gestione in quanto è necessario eseguire il provisioning di cluster e VPC aggiuntivi per soddisfare i requisiti di scalabilità. Consultare Kubernetes Scalability thresholds su GitHub per ulteriori informazioni.

Compatibilità con le funzionalità del plug-in CNI di Amazon VPC per Kubernetes

È possibile utilizzare prefissi IP con le seguenti caratteristiche:

- Traslazione degli indirizzi di rete di origine IPv4: Per ulteriori informazioni, consulta the section called "Traffico in uscita".
- Indirizzi IPv6 verso cluster, pod e servizi: Per ulteriori informazioni, consulta the section called "IPv6".
- Limitazione del traffico tramite le policy di rete di Kubernetes: Per ulteriori informazioni, consulta the section called "Policy di Kubernetes".

L'elenco seguente fornisce informazioni sulle impostazioni del plug-in CNI di Amazon VPC applicabili. Per maggiori informazioni su ciascuna impostazione, consultare amazon-vpc-cni-k8s su GitHub.

- `WARM_IP_TARGET`
- `MINIMUM_IP_TARGET`
- `WARM_PREFIX_TARGET`

Considerazioni

Quando si utilizza questa funzionalità, considerare i seguenti aspetti:

- Ogni tipo di istanza Amazon EC2 supporta un numero massimo di Pod. Se il gruppo di nodi gestito è costituito da più tipi di istanza, a tutti i nodi del cluster viene applicato il numero minimo di Pod massimo per un'istanza del cluster.
- Per impostazione predefinita, il numero massimo di `Pods` che puoi eseguire su un nodo è pari a 110, ma puoi modificare questo numero. Se cambi il numero e disponi di un gruppo di nodi gestiti, il prossimo aggiornamento dell'AMI o del modello di avvio del gruppo di nodi comporta la creazione di nuovi nodi con il valore modificato.
- Quando passi dall'assegnazione di indirizzi IP all'assegnazione di prefissi IP, si consiglia di creare nuovi gruppi di nodi per aumentare il numero di indirizzi IP disponibili, anziché effettuare una sostituzione progressiva dei nodi esistenti. L'esecuzione di Pod su un nodo a cui sono stati assegnati sia indirizzi IP che prefissi può portare a un'incoerenza nella capacità degli indirizzi IP pubblicizzati, con un impatto sui carichi di lavoro futuri sul nodo. Per il metodo consigliato di eseguire la transizione, consulta la modalità Prefix Delegation per Linux in Amazon EKS Best Practices Guide.
- L'ambito del gruppo di sicurezza è a livello di nodo: Per ulteriori informazioni, consulta Gruppo di sicurezza.
- I prefissi IP assegnati a un'interfaccia di rete supportano un'elevata densità di Pod per nodo e offrono il miglior tempo di avvio.
- I prefissi IP e gli indirizzi IP sono associati alle interfacce di rete elastiche Amazon EC2 standard. Ai pod che richiedono gruppi di sicurezza specifici viene assegnato l'indirizzo IP primario di un'interfaccia di rete di filiali. È possibile combinare i Pod ottenendo gli indirizzi IP o gli indirizzi IP dai prefissi IP con i Pod che ottengono interfacce di rete di filiali sullo stesso nodo.
- Solo per i cluster con nodi Linux.
  - Dopo aver configurato il componente aggiuntivo per assegnare prefissi alle interfacce di rete, non è possibile eseguire il downgrade del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes a una versione inferiore a `1.9.0` (o `1.10.1`) senza rimuovere tutti i nodi in tutti i gruppi di nodi nel cluster.
  - Se si utilizzano anche gruppi di sicurezza per i Pod, con `POD_SECURITY_GROUP_ENFORCING_MODE=standard` e `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, quando i Pod comunicano con endpoint esterni al VPC, vengono utilizzati i gruppi di sicurezza del nodo, anziché i gruppi di sicurezza assegnati ai Pod.

Se si utilizzano anche gruppi di sicurezza per Pod, con `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, quando i `Pods` comunicano con endpoint esterni al VPC, vengono utilizzati i gruppi di sicurezza di `Pod's`.

## Aumentare gli indirizzi IP disponibili per il nodo Amazon EKS

Puoi aumentare sensibilmente il numero di indirizzi IP che i nodi possono assegnare ai Pod assegnando prefissi IP ai nodi, anziché assegnare singoli indirizzi IP secondari.

### Prerequisiti

- Necessiti di un cluster esistente. Per implementarne uno, consulta the section called "Creazione di un cluster ".
- Le sottoreti in cui si trovano i nodi Amazon EKS devono avere sufficienti blocchi contigui (per i cluster) o `/28` (per `IPv4` i cluster) Inter-Domain Classless `/80` Routing `IPv6` (CIDR). Puoi disporre solo di nodi Linux in un cluster `IPv6`. L'uso dei prefissi IP può non riuscire se gli indirizzi IP sono sparsi nella sottorete CIDR. Consigliamo quanto segue:
  - L'utilizzo di una prenotazione di sottorete CIDR in modo che, anche se alcuni indirizzi IP all'interno dell'intervallo riservato sono ancora in uso, al momento del rilascio, gli indirizzi IP non vengano riassegnati. Ciò assicura che i prefissi siano disponibili per l'assegnazione senza segmentazione.
  - Usa nuove sottoreti utilizzate specificamente per l'esecuzione di carichi di lavoro a cui sono assegnati i prefissi IP. I carichi di lavoro sia di Windows sia di Linux possono essere eseguiti nella stessa sottorete quando si assegnano prefissi IP.
- Per assegnare prefissi IP ai tuoi nodi, i tuoi nodi devono esserlo. AWS Nitro-based Le istanze che non lo sono Nitro-based continuano ad allocare singoli indirizzi IP secondari, ma hanno un numero significativamente inferiore di indirizzi IP da assegnare ai Pods rispetto alle istanze. Nitro-based
- Solo per cluster con nodi Linux`IPv4`: se il cluster è configurato per la famiglia, devi avere installata la versione `1.9.0` o una versione successiva del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes. Puoi controllare la versione attuale con il seguente comando.

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

Se il cluster è configurato per la famiglia `IPv6`, è necessario che sia installata la versione `1.10.1` del componente aggiuntivo. Se la versione del plugin è precedente a quelle richieste, è necessario

aggiornarla. Per ulteriori informazioni, consulta le sezioni sugli aggiornamenti di Assign IPs to Pods with the Amazon VPC CNI.

- Solo per i cluster con nodi Windows
  - Per il cluster, devi abilitare il supporto Windows. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

Assegnare i prefissi degli indirizzi IP ai nodi

Configurare il cluster per assegnare i prefissi degli indirizzi IP ai nodi. Completare la procedura corrispondente al sistema operativo del nodo.

Linux

1. Abilita il parametro per assegnare prefissi alle interfacce di rete per Amazon VPC CNI. DaemonSet Quando implementi un cluster, la versione `1.10.1` o successiva del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes viene implementata con esso. Se crei il cluster con la famiglia `IPv6`, questa impostazione era su `true` di default. Se crei il cluster con la famiglia `IPv4`, questa impostazione era su `false` di default.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
```

> **Important**
>
> Anche se la sottorete dispone di indirizzi IP disponibili, se non dispone di blocchi `/28` contigui disponibili, vedrai il seguente errore nei log del plug-in di Amazon VPC CNI per Kubernetes.
>
> ```
> InsufficientCidrBlocks: The specified subnet does not have enough free cidr
>  blocks to satisfy the request
> ```
>
> Ciò può verificarsi a causa della frammentazione degli indirizzi IP secondari esistenti distribuiti in una sottorete. Per risolvere questo errore, creare una nuova sottorete e avviare i Pod lì oppure utilizzare una prenotazione CIDR sottorete Amazon EC2 per riservare spazio all'interno di una sottorete da utilizzare con l'assegnazione del prefisso. Per ulteriori informazioni, consulta la sezione relativa a Prenotazioni della CIDR per la sottorete nella Guida per l'utente di Amazon VPC.

2. Se intendi implementare un gruppo di nodi gestito senza un modello di avvio o con uno in cui non è specificato un ID AMI e se utilizzi una versione del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes corrispondente o successiva alle versioni riportate nei prerequisiti, vai al passaggio successivo. I gruppi di nodi gestiti calcolano automaticamente il numero massimo di Pod per te.

   Se stai implementando un gruppo di nodi autogestito o un gruppo di nodi gestiti con un modello di avvio in cui hai specificato un ID AMI, devi impostare il numero massimo di pod per i tuoi nodi. Per ulteriori informazioni su come determinare il valore appropriato, consulta. the section called "Come viene determinato MaxPods"

   > **Important**
   >
   > I gruppi di nodi gestiti impongono un numero massimo al valore di`maxPods`. Per le istanze con meno di 30 vCPUs il numero massimo è 110 e per tutte le altre istanze il numero massimo è 250. Questo numero massimo viene applicato indipendentemente dal fatto che la delega del prefisso sia abilitata o meno.

3. Se usi un cluster configurato per `IPv6`, vai alla Passaggio successiva.

   È possibile specificare i parametri una delle seguenti opzioni. Per determinare quale opzione è giusta per te e quale valore fornirle, vedi WARM_PREFIX_TARGET, WARM_IP_TARGET e MINIMUM_IP_TARGET su. GitHub

   Puoi sostituire i valori di esempio con un valore maggiore di zero.

   - `WARM_PREFIX_TARGET`

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - `WARM_IP_TARGET` o `MINIMUM_IP_TARGET` – Se uno dei due valori è impostato, sovrascrive qualsiasi valore impostato per `WARM_PREFIX_TARGET`.

     ```
     kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
     ```

     ```
     kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
     ```

4. Crea uno dei seguenti tipi di gruppi di nodi con almeno un tipo di istanza Amazon EC2 Nitro Amazon Linux 2023. Per un elenco dei tipi di istanze Nitro, consulta Instances built on the

Nitro System nella Guida per l'utente di Amazon EC2. Questa funzionalità non è supportata da Windows. Per le opzioni che includono *110*, sostituirlo con il valore del passaggio 3 (consigliato) o con il proprio valore.

- Self-managed— Implementa il gruppo di nodi utilizzando le istruzioni in Creazione di nodi Amazon Linux autogestiti. Prima di creare lo CloudFormation stack, apri il file modello e regolalo `UserData` in `NodeLaunchTemplate` modo che sia come segue.

```
...
            apiVersion: node.eks.aws/v1alpha1
            kind: NodeConfig
            spec:
              cluster:
                name: ${ClusterName}
                apiServerEndpoint: ${ApiServerEndpoint}
                certificateAuthority: ${CertificateAuthorityData}
                cidr: ${ServiceCidr}
              kubelet:
                config:
                  maxPods: 110
...
```

Se stai utilizzando `eksctl` per creare il gruppo di nodi, puoi usare il seguente comando.

```
eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node
 110
```

- Gestito: implementare il gruppo di nodi utilizzando una delle opzioni seguenti:
  - Senza un modello di avvio o con un modello di avvio senza un ID AMI specificato: completare la procedura in Create a managed node group for your cluster. I gruppi di nodi gestiti calcolano automaticamente il `max-pods` valore consigliato da Amazon EKS per te.
  - Con un modello di avvio con un ID AMI specificato – nel modello di lancio, specificare un ID AMI ottimizzato per Amazon EKS o un'AMI personalizzata sviluppata con l'AMI ottimizzato Amazon EKS, quindi implementare il gruppo di nodi utilizzando un modello di avvio e fornire i seguenti dati utente nel modello di avvio. Questi dati utente trasmettono un `NodeConfig` oggetto che viene letto dallo `nodeadm` strumento sul nodo. Per ulteriori informazioni`nodeadm`, consulta la documentazione di nodeadm.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="//"
```

```
--//
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
 cluster:
   apiServerEndpoint: my-cluster
   certificateAuthority: LS0t...
   cidr: 10.100.0.0/16
   name: my-cluster
 kubelet:
   config:
     maxPods: 110
--//--
```

Se stai utilizzando `eksctl` per creare il gruppo di nodi, puoi usare il seguente comando.

```
eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
```

Se hai creato un'AMI personalizzata che non è stata sviluppata con quella ottimizzata per Amazon EKS, devi creare autonomamente la configurazione.

> **Note**
>
> Se desideri assegnare anche indirizzi IP ai Pod di una sottorete diversa da quella dell'istanza, devi abilitare la funzionalità in questa Passaggio. Per ulteriori informazioni, consulta the section called "Rete personalizzata".

Windows

1. Abilita l'assegnazione di prefissi IP.
   a. Apri `amazon-vpc-cni ConfigMap` per la modifica.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Aggiungi la seguente riga al file `data`:

```
  enable-windows-prefix-delegation: "true"
```

c. Salva il file e chiudi l'editor.

d. Accertati che la riga sia stata aggiunta a `ConfigMap`.

```
kubectl get configmap -n kube-system amazon-vpc-cni -o "jsonpath={.data.enable-
windows-prefix-delegation}"
```

Se l'output restituito non è `true`, potrebbe essersi verificato un errore. Prova a completare di nuovo il passaggio.

> **Important**
>
> Anche se la sottorete dispone di indirizzi IP disponibili, se non dispone di blocchi `/28` contigui disponibili, vedrai il seguente errore nei log del plug-in di Amazon VPC CNI per Kubernetes.
>
> ```
> InsufficientCidrBlocks: The specified subnet does not have enough free cidr
>  blocks to satisfy the request
> ```
>
> Ciò può verificarsi a causa della frammentazione degli indirizzi IP secondari esistenti distribuiti in una sottorete. Per risolvere questo errore, creare una nuova sottorete e avviare i Pod lì oppure utilizzare una prenotazione CIDR sottorete Amazon EC2 per riservare spazio all'interno di una sottorete da utilizzare con l'assegnazione del prefisso. Per ulteriori informazioni, consulta la sezione relativa a Prenotazioni della CIDR per la sottorete nella Guida per l'utente di Amazon VPC.

2. (Facoltativo) Specifica una configurazione aggiuntiva per controllare il comportamento di prescalabilità e scalabilità dinamica del cluster. Per ulteriori informazioni, vedere Opzioni di configurazione con modalità di delega dei prefissi su Windows attivo. GitHub

a. Apri `amazon-vpc-cni` `ConfigMap` per la modifica.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

b. Sostituisci i valori di esempio con un valore maggiore di zero e aggiungi le voci necessarie alla `data` sezione del. `ConfigMap` Se imposti un valore per `warm-ip-target` o `minimum-ip-target`, il valore sostituisce qualsiasi valore impostato per `warm-prefix-target`.

```
  warm-prefix-target: "1"
  warm-ip-target: "5"
  minimum-ip-target: "2"
```

c. Salva il file e chiudi l'editor.

3. Creare gruppi di nodi di Windows con almeno un tipo di istanza Amazon EC2 Nitro. Per un elenco dei tipi di istanze Nitro, consultare Instances built on the Nitro System nella Guida per l'utente di Amazon EC2. Per impostazione predefinita, il numero massimo di Pod che è possibile distribuire su un nodo è pari a 110. Se vuoi aumentare o diminuire questo numero, specifica quanto segue nei dati utente per la configurazione di bootstrap. Sostituisci *max-pods-quantity* con il tuo valore di pod massimo.

```
-KubeletExtraArgs '--max-pods=max-pods-quantity'
```

Se si sta implementando gruppi di nodi gestiti, questa configurazione deve essere aggiunta nel modello di avvio. Per ulteriori informazioni, consulta the section called "Modelli di avvio". Per ulteriori informazioni sui parametri di configurazione dello script bootstrap di Windows, consultare the section called "Parametri di configurazione dello script di bootstrap".

Determinare il numero massimo di pod e gli indirizzi IP disponibili

1. Dopo aver implementato i nodi, visualizzare i nodi del cluster.

```
kubectl get nodes
```

Di seguito viene riportato un output di esempio:

```
NAME                                             STATUS     ROLES    AGE   VERSION
ip-192-168-22-103.region-code.compute.internal   Ready      <none>   19m   v1.XX.X-
eks-6b7464
ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-
eks-6b7464
```

2. Descrivi uno dei nodi per determinare il valore di `max-pods` per il nodo e il numero di indirizzi IP disponibili. Sostituisci *192.168.30.193* con l'indirizzo `IPv4` nel nome di uno dei tuoi nodi restituiti nell'output del passaggio precedente.

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|
PrivateIPv4Address'
```

Di seguito viene riportato un output di esempio.

```
pods:                                   110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

Nell'output precedente, `110` è il numero massimo di pod che Kubernetes distribuirà nel nodo, anche se *144* gli indirizzi IP sono disponibili.

Assegnazione dei gruppi di sicurezza ai singoli pod

Si applica a: nodi Linux con istanze Amazon EC2

Si applica a: sottoreti private

I gruppi di sicurezza per i pod integrano i gruppi di sicurezza Amazon EC2 con i pod Kubernetes. È possibile utilizzare i gruppi di sicurezza Amazon EC2 per definire regole che consentono il traffico di rete in entrata e in uscita da e verso i pod implementati ai nodi in esecuzione su molti tipi di istanze Amazon EC2 e Fargate. Per una descrizione dettagliata di questa funzionalità, consulta il post di blog Introduzione ai gruppi di sicurezza per i pod.

Compatibilità con il plugin CNI di Amazon VPC per le funzionalità di Kubernetes

È possibile utilizzare gruppi di sicurezza per pod con le seguenti funzionalità:

- Traslazione degli indirizzi di rete di origine IPv4: per ulteriori informazioni, consulta the section called "Traffico in uscita".
- Indirizzi IPv6 verso cluster, pod e servizi: per ulteriori informazioni, consulta the section called "IPv6".
- Limitazione del traffico tramite le policy di rete di Kubernetes: per ulteriori informazioni, consulta the section called "Policy di Kubernetes".

Considerazioni

Prima di implementare i gruppi di sicurezza per i pod, considera i limiti e le condizioni seguenti:

- I gruppi di sicurezza per Pods non possono essere utilizzati con i nodi Windows o la modalità automatica EKS.
- I gruppi di sicurezza per i pod possono essere utilizzati con cluster configurati per la famiglia `IPv6` che contengono i nodi Amazon EC2 utilizzando la versione 1.16.0 o successive del plugin CNI di Amazon VPC. Puoi utilizzare i gruppi di sicurezza per Pods con cluster configurati per la `IPv6` famiglia che contengono solo nodi Fargate utilizzando la versione 1.7.7 o successiva del plug-in Amazon VPC CNI. Per ulteriori informazioni, consulta the section called "IPv6".
- I gruppi di sicurezza per Pods sono supportati dalla maggior parte delle famiglie di istanze Nitro-based Amazon EC2, ma non da tutte le generazioni di una famiglia. Ad esempio, sono supportate la famiglia e le generazioni dell'istanza `m5`, `c5`, `r5`, `m6g`, `c6g` e `r6g`. Non è supportato alcun tipo di istanza nella famiglia `t`. Per un elenco completo dei tipi di istanze supportati, consulta il file limits.go su. GitHub I nodi devono essere uno dei tipi di istanza elencati che contengono `IsTrunkingCompatible: true` nel file.
- Se si utilizzano allo stesso tempo una rete personalizzata e gruppi di sicurezza per i pod, il gruppo di sicurezza specificato dai gruppi di sicurezza per pod è utilizzato in alternativa al gruppo di sicurezza specificato in `ENIConfig`.
- Se si utilizza la versione `1.10.2` o precedente del plugin CNI di Amazon VPC e si include l'impostazione `terminationGracePeriodSeconds` nelle specifiche del pod, il valore per l'impostazione non può essere zero.
- Se si utilizza la versione `1.10` o precedente del plugin CNI di Amazon VPC o la versione `1.11` con l'impostazione predefinita `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, i servizi Kubernetes di tipo `NodePort` e `LoadBalancer` che utilizzano le destinazioni di istanza con `externalTrafficPolicy` impostato su `Local` non sono supportati con i pod a cui si assegnano i gruppi di sicurezza. Per ulteriori informazioni sull'utilizzo di un load balancer con target di istanza, consultare the section called "Bilanciamento del carico di rete".
- La fonte NAT è disabilitata per il traffico in uscita dai pod con gruppi di sicurezza assegnati, in modo che siano applicate le regole dei gruppi di sicurezza in uscita, se si utilizza la versione `1.10` o precedente del plugin CNI di Amazon VPC o la versione `1.11` con l'impostazione predefinita `POD_SECURITY_GROUP_ENFORCING_MODE=strict`. Per accedere a Internet, è necessario avviare pod con gruppi di sicurezza assegnati su nodi distribuiti in una sottorete privata configurata con un gateway NAT o un'istanza. I pod con gruppi di sicurezza assegnati distribuiti alle sottoreti pubbliche non sono in grado di accedere a Internet.

  Se si utilizza la versione `1.11` o successiva del plugin con `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, il traffico del pod destinato all'esterno

del VPC è convertito nell'indirizzo IP dell'interfaccia di rete primaria dell'istanza. Per questo traffico sono utilizzate le regole nei gruppi di sicurezza per l'interfaccia di rete primaria, anziché le regole nei gruppi di sicurezza dei pod.

- Per utilizzare le policy di rete Calico con pod associati ai gruppi di sicurezza, è necessario utilizzare la versione `1.11.0` o successiva del plugin CNI di Amazon VPC e impostare `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. In caso contrario, i flussi di traffico da e verso i pod con gruppi di sicurezza associati non sono soggetti ad esecuzione su policy di rete Calico e la loro applicazione è esclusivamente limitata al gruppo di sicurezza Amazon EC2. Per aggiornare la versione di CNI di Amazon VPC, consulta la sezione the section called "CNI di Amazon VPC".
- I pod in esecuzione su nodi Amazon EC2 che utilizzano gruppi di sicurezza in cluster che utilizzano NodeLocal DNSCache sono supportati solo con la versione `1.11.0` o successiva del plug-in Amazon VPC CNI e con =. `POD_SECURITY_GROUP_ENFORCING_MODE` `standard` Per aggiornare la versione del plug-in CNI di Amazon VPC, consulta la sezione the section called "CNI di Amazon VPC".
- I gruppi di sicurezza per pod potrebbero portare a un aumento della latenza di avvio dei pod per quelli con un alto tasso di abbandono. Ciò è dovuto alla limitazione della velocità nel controller delle risorse.
- L'ambito del gruppo di sicurezza EC2 è in Pod-level - Per ulteriori informazioni, consulta Security group. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html

  Se hai impostato `POD_SECURITY_GROUP_ENFORCING_MODE=standard` e `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, il traffico destinato agli endpoint esterni al VPC utilizza i gruppi di sicurezza del nodo, non i gruppi di sicurezza del pod.

## Configurazione del plugin CNI di Amazon VPC per Kubernetes per gruppi di sicurezza per i pod Amazon EKS

Se si utilizzano pod con istanze Amazon EC2, devi configurare il plugin CNI di Amazon VPC per Kubernetes per gruppi di sicurezza

Se si utilizzano solo i pod di Fargate e non si dispone di nodi Amazon EC2 nel cluster, consulta the section called "SecurityGroupPolicy".

1. Controlla la versione corrente del plugin CNI di Amazon VPC per Kubernetes con il comando seguente:

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Di seguito viene riportato un output di esempio.

```
v1.7.6
```

Se la versione del plugin CNI di Amazon VPC per Kubernetes è precedente a `1.7.7`, aggiorna il plugin alla versione `1.7.7` o successiva. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

2. Aggiungi la policy IAM gestita AmazonEKSVPCResourceController al ruolo del cluster associato al cluster Amazon EKS. La policy consente al ruolo di gestire le interfacce di rete e i relativi indirizzi IP privati e di collegare e scollegare le interfacce di rete da e verso le istanze.
   a. Recupera il nome del ruolo IAM del cluster e archivialo in una variabile. Sostituisci *my-cluster* con il nome del cluster.

   ```
   cluster_role=$(aws eks describe-cluster --name my-cluster --query cluster.roleArn
    --output text | cut -d / -f 2)
   ```

   b. Collegare la policy al ruolo.

   ```
   aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
   AmazonEKSVPCResourceController --role-name $cluster_role
   ```

3. Abilita il componente aggiuntivo Amazon VPC CNI per gestire le interfacce di rete per Pods impostando la variabile su in. `ENABLE_POD_ENI` `true` `aws-node` DaemonSet Una volta impostato questo parametro su `true`, per ciascun nodo nel cluster il componente aggiuntivo crea una risorsa `cninode` personalizzata. Il controller di risorse VPC crea e allega un'interfaccia di rete speciale chiamata interfaccia di rete trunk con la descrizione `aws-k8s-trunk-eni`.

   ```
   kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
   ```

   > **Note**
   >
   > L'interfaccia di rete trunk è inclusa nel numero massimo di interfacce di rete supportate dal tipo di istanza. Per un elenco del numero massimo di interfacce di rete supportate da ciascun tipo di istanza, consulta Indirizzi IP per interfaccia di rete per tipo di istanza

nella Guida per l'utente di Amazon EC2. Se il nodo ha già allegato il numero massimo di interfacce di rete standard, il controller di risorse VPC riserverà uno spazio. Sarà necessario ridurre verticalmente i pod in esecuzione abbastanza da consentire al controller di scollegare ed eliminare un'interfaccia di rete standard, creare l'interfaccia di rete trunk e allegarla all'istanza.

4. Con il seguente comando è possibile vedere quale nodo ha una risorsa `CNINode` personalizzata. Se viene restituito l'output `No resources found`, attendere alcuni secondi e riprovare. La fase precedente richiede il riavvio del plugin CNI di Amazon VPC per pod Kubernetes, che richiede alcuni secondi.

```
kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

Se si utilizzano versioni VPC CNI precedenti alla `1.15`, sono state utilizzate le etichette dei nodi anziché la risorsa `CNINode` personalizzata. Con il seguente comando, è possibile vedere quale nodo ha l'etichetta del nodo `aws-k8s-trunk-eni` impostata su `true`. Se viene restituito l'output `No resources found`, attendere alcuni secondi e riprovare. La fase precedente richiede il riavvio del plugin CNI di Amazon VPC per pod Kubernetes, che richiede alcuni secondi.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
```

Una volta creata l'interfaccia di rete trunk, ai pod possono essere assegnati indirizzi IP secondari dalla rete trunk o dalle interfacce di rete standard. L'interfaccia trunk viene eliminata automaticamente se il nodo viene eliminato.

Quando si distribuisce un gruppo di sicurezza per un pod in un passaggio successivo, il controller di risorse VPC crea un'interfaccia di rete speciale denominata interfaccia di rete di filiali con una descrizione di `aws-k8s-branch-eni` e vi associa i gruppi di sicurezza. Le interfacce di rete di filiali vengono create in aggiunta alle interfacce di rete standard e trunk allegate al nodo.

Se si utilizzano sonde liveness o readiness, è inoltre necessario disabilitare il demultiplexing precoce TCP, in modo che `kubelet` possa connettersi a pod su interfacce di rete di filiali tramite TCP. Per disabilitare demultiplexing precoce TCP, esegui il comando seguente:

```
kubectl patch daemonset aws-node -n kube-system \
```

```
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-init"}]}}}}'
```

> **Note**
>
> Se non utilizzi il plugin CNI di Amazon VPC `1.11.0` o successivo per il componente aggiuntivo di Kubernetes e impostarlo `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, come descritto nel passaggio successivo, non devi eseguire il comando precedente.

5. Se il cluster utilizza `NodeLocal DNSCache` o desideri utilizzare la policy di rete Colico con i pod che dispongono dei propri gruppi di sicurezza, o se disponi di servizi Kubernetes di tipo `NodePort` e `LoadBalancer` che utilizzano le destinazioni di istanza con `externalTrafficPolicy` impostato su `Local` per pod a cui desideri assegnare i gruppi di sicurezza, dovrai utilizzare la versione `1.11.0` o successiva del plugin CNI di Amazon VPC per il componente aggiuntivo di Kubernetes e dovrai abilitare la seguente impostazione:

   ```
   kubectl set env daemonset aws-node -n kube-system
    POD_SECURITY_GROUP_ENFORCING_MODE=standard
   ```

   IMPORTANTE: le regole del gruppo di sicurezza dei pod non si applicano al traffico tra pod o tra pod e servizi, esempio **`kubelet`** o **`nodeLocalDNS`**, che si trovano sullo stesso nodo. I pod che utilizzano gruppi di sicurezza differenti sullo stesso nodo non possono comunicare perché sono configurati in sottoreti diverse e l'instradamento è disabilitato tra tali sottoreti. Traffico in uscita dai pod verso gli indirizzi esterni al VPC è l'indirizzo di rete convertito nell'indirizzo IP dell'interfaccia di rete primaria dell'istanza (a meno che non sia stato impostato `AWS_VPC_K8S_CNI_EXTERNALSNAT=true`). Per questo traffico sono utilizzate le regole nei gruppi di sicurezza per l'interfaccia di rete primaria, anziché le regole nei gruppi di sicurezza dei pod. ** Affinché questa impostazione si applichi ai pod esistenti, è necessario riavviare i pod o i nodi su cui i pod sono in esecuzione.

6. Per scoprire come utilizzare una politica di gruppo di sicurezza per il pod, consulta the section called "SecurityGroupPolicy".

Utilizzo di una policy di gruppo di sicurezza per un pod Amazon EKS

Per utilizzare i gruppi di sicurezza per i pod, è necessario disporre di un gruppo di sicurezza esistente. I passaggi seguenti mostrano l'utilizzo delle policy di gruppo di sicurezza per un pod. Se

non diversamente indicato, completa tutti i passaggi nello stesso terminale, poiché nei passaggi successivi sono utilizzate variabili che non persistono tra i terminali.

Se disponi di un Pod con istanze Amazon EC2, devi configurare il plugin prima di utilizzare questa procedura. Per ulteriori informazioni, consulta the section called "Configura".

1. Crea un namespace Kubernetes in cui distribuire le risorse . È possibile sostituire *my-namespace* con il nome di uno spazio dei nomi che si desidera utilizzare.

```
kubectl create namespace my-namespace
```

2. Implementa una `SecurityGroupPolicy` Amazon EKS nel cluster.

   a. Copia i seguenti contenuti sul dispositivo. Puoi sostituirlo *podSelector* con `serviceAccountSelector` se preferisci selezionare i Pod in base alle etichette degli account di servizio. É necessario specificare uno selettore. Un `podSelector` vuoto (ad esempio: `podSelector: {}`) seleziona tutti i pod nel namespace. Puoi *my-role* cambiare il nome del tuo ruolo. Un `serviceAccountSelector` vuoto seleziona tutti gli account di servizio nel namespace. Puoi sostituirlo *my-security-group-policy* con un nome per te `SecurityGroupPolicy` e *my-namespace* con lo spazio dei nomi in cui desideri crearlo. `SecurityGroupPolicy`

   È necessario sostituirlo *my_pod_security_group_id* con l'ID di un gruppo di sicurezza esistente. Se non disponi ancora di un gruppo di sicurezza dovrai crearne uno. Per ulteriori informazioni, consulta Gruppi di sicurezza Amazon EC2 per le istanze Linux nella Guida per l'utente di Amazon EC2. È possibile specificare 1-5 gruppi IDs di sicurezza. Se si specificano più ID, la combinazione di tutte le regole in tutti i gruppi di sicurezza sarà valida per i pod selezionati.

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
```

```
      - my_pod_security_group_id
EOF
```

> **Important**
>
> Il gruppo di sicurezza o i gruppi specificati per i pod devono soddisfare i criteri seguenti:
>
> - Devono essere esistenti. Se non lo sono, quando si distribuisce un pod che corrisponde al selettore, il pod rimane bloccato nel processo di creazione. Se si descrive il pod, si otterrà un messaggio di errore simile al seguente: `An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-05b1d815d1EXAMPLE' does not exist.`
> - Devono consentire la comunicazione in ingresso dal gruppo di sicurezza applicato ai nodi (per `kubelet`) su tutte le porte per cui sono state configurate le sonde.
> - Devono consentire la comunicazione in uscita tramite le porte 53 `TCP` e `UDP` a un gruppo di sicurezza assegnato ai pod (o ai nodi su cui sono eseguiti i pod) che eseguono CoreDNS. Il gruppo di sicurezza per i pod CoreDNS deve consentire il traffico in ingresso tramite le porte 53 `TCP` e `UDP` dal gruppo di sicurezza specificato.
> - Devono disporre delle regole in entrata e in uscita necessarie per comunicare con altri pod.
> - Devono disporre di regole che consentano ai pod di comunicare con il piano di controllo Kubernetes se si utilizza il gruppo di sicurezza con Fargate. Il modo più semplice per eseguire questa operazione consiste nello specificare il gruppo di sicurezza del cluster come uno dei gruppi di sicurezza.
>
> Le policy dei gruppi di sicurezza si applicano solo ai nuovi pod pianificati. Non incidono sui pod in esecuzione.

b. Implementare la policy.

```
kubectl apply -f my-security-group-policy.yaml
```

3. Implementa un'applicazione di esempio con un'etichetta che corrisponda al valore *`my-role`* per *`podSelector`* specificato nel passaggio precedente.

   a. Copia i seguenti contenuti sul dispositivo. Sostituisci i valori di esempio con i tuoi, quindi esegui il comando modificato. Se lo sostituisci*`my-role`*, assicurati che sia uguale al valore specificato per il selettore in un passaggio precedente.

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. Implementare l'applicazione con il seguente comando. Quando si implementa l'applicazione, il plugin CNI di Amazon VPC per Kubernetes corrisponde all'etichetta `role` e i gruppi di sicurezza specificati nel passaggio precedente sono applicati al pod.

```
kubectl apply -f sample-application.yaml
```

4. Visualizza i pod implementati con l'applicazione esemplificativa. Nella parte restante di questo argomento, questo terminale è indicato come `TerminalA`.

```
kubectl get pods -n my-namespace -o wide
```

Di seguito viene riportato un output di esempio.

```
NAME                              READY   STATUS    RESTARTS   AGE     IP
  NODE                                                NOMINATED NODE   READINESS GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s   192.168.70.145
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s   192.168.73.207
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

> **Note**
>
> Prova questi suggerimenti se qualche pod è bloccato.
>
> - Se qualche pod è bloccato nello stato `Waiting`, esegui `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* . Se visualizzi `Insufficient permissions: Unable to create Elastic Network Interface.`, verifica di avere aggiunto la policy IAM al ruolo cluster IAM in un passaggio precedente.
> - Se qualche pod è bloccato nello stato `Pending`, verifica che il tipo di istanza del nodo sia elencato in limits.go e che non sia già stato raggiunto il prodotto del numero massimo di interfacce di rete di filiali supportate dal tipo di istanza moltiplicato per il numero di nodi nel gruppo di nodi. Ad esempio, una istanza `m5.large` supporta nove interfacce di rete di filiali. Se il gruppo di nodi dispone di cinque nodi, è possibile creare un massimo di 45 interfacce di rete di filiali per il gruppo di nodi. Il 46° pod che si tenta di

implementare si troverà in uno stato di `Pending` finché non viene eliminato un altro pod a cui sono associati gruppi di sicurezza.

Se si esegue `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* e si visualizza un messaggio simile al seguente, può essere ignorato in modo sicuro. Questo messaggio potrebbe essere visualizzato quando il plugin CNI di Amazon VPC per Kubernetes tenta di impostare la rete host e fallisce durante la creazione dell'interfaccia di rete. Il plug-in registra questo evento fino a quando non viene creata l'interfaccia di rete.

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f"
 network for Pod "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace" network:
 add cmd: failed to assign an IP address to container
```

Non è possibile superare il numero massimo di pod eseguibili sul tipo di istanza. Per un elenco del numero massimo di Pod che puoi eseguire su ogni tipo di istanza, consulta eni-max-pods.txt on. GitHub Quando si elimina un pod a cui sono associati gruppi di sicurezza o si elimina il nodo su cui è in esecuzione il pod, il controller di risorse VPC elimina l'interfaccia di rete di filiali. Se elimini un cluster con pod che utilizzano pod per gruppi di sicurezza, il controller non elimina le interfacce di rete delle diramazioni, per cui dovrai eliminarle personalmente. Per ulteriori informazioni sull'eliminazione delle interfacce di rete, consulta Eliminazione di un'interfaccia di rete nella Guida per l'utente Amazon EC2.

5. In un terminale separato, accedi alla shell di uno dei pod. Nella parte restante di questo argomento, questo terminale è indicato come `TerminalB`. Sostituiscilo *`5df6f7687b-4fbjm`* con l'ID di uno dei Pod restituiti nell'output del passaggio precedente.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. Confermare il funzionamento dell'applicazione di esempio dalla shell in `TerminalB`.

```
curl my-app
```

Di seguito viene riportato un output di esempio.

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Welcome to nginx!</title>
[...]
```

L'output è stato ricevuto in quanto tutti i pod che eseguono l'applicazione sono associati al gruppo di sicurezza creato. Tale gruppo contiene una regola che abilita il traffico tra tutti i pod a cui è associato il gruppo di sicurezza. Il traffico DNS è consentito in uscita da tale gruppo di sicurezza al gruppo di sicurezza del cluster associato ai nodi. I nodi eseguono i pod CoreDNS, su cui i pod hanno eseguito la ricerca del nome.

7. Da `TerminalA`, rimuovere le regole del gruppo di sicurezza che consentono la comunicazione DNS dal gruppo di sicurezza al gruppo di sicurezza del cluster. Se non hai aggiunto le regole DNS al gruppo di sicurezza del cluster in un passaggio precedente, sostituiscile *`$my_cluster_security_group_id`* con l'ID del gruppo di sicurezza in cui hai creato le regole.

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. Tentare di accedere nuovamente all'applicazione da `TerminalB`.

```
curl my-app
```

Di seguito viene riportato un output di esempio.

```
curl: (6) Could not resolve host: my-app
```

Il tentativo ha esito negativo perché il pod non è più in grado di accedere ai pod CoreDNS a cui è associato il gruppo di sicurezza del cluster. Il gruppo di sicurezza del cluster non dispone più delle relative regole che consentono la comunicazione DNS dal gruppo di sicurezza associato al pod.

Se si tenta di accedere all'applicazione utilizzando gli indirizzi IP restituiti in un passaggio precedente per uno dei pod, si riceve comunque una risposta perché tutte le porte sono consentite tra i pod a cui è associato il gruppo di sicurezza e non è necessaria una ricerca del nome.

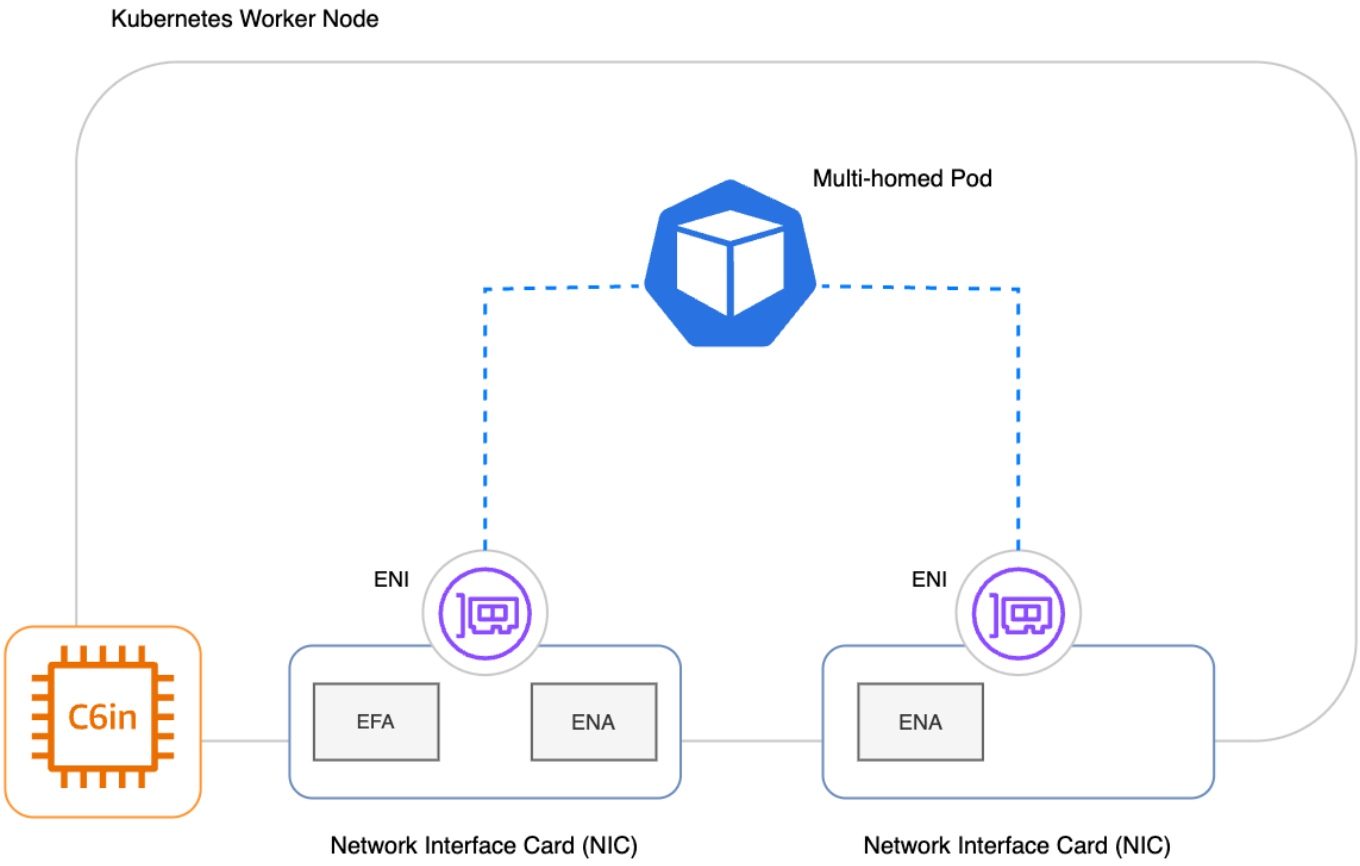
9. Dopo avere eseguito diverse prove, è possibile rimuovere la policy del gruppo di sicurezza, l'applicazione e il gruppo di sicurezza di esempio creati. Esegui i comandi seguenti da `TerminalA`.

```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

## Collegamento di interfacce di rete multiple ai pod

Per impostazione predefinita, il plugin CNI di Amazon VPC assegna un indirizzo IP a ciascun pod. Questo indirizzo IP è collegato a un'interfaccia di rete elastica che gestisce tutto il traffico in entrata e in uscita per il pod. Per aumentare la larghezza di banda e le prestazioni della velocità dei pacchetti al secondo, puoi utilizzare la  Multi-NIC funzionalità  del VPC CNI per configurare un pod multi-homed. Un pod multi-homed è un singolo pod Kubernetes che utilizza più interfacce di rete (e più indirizzi IP). Eseguendo un pod multi-homed, è possibile distribuire il traffico delle applicazioni su più interfacce di rete utilizzando connessioni simultanee. Ciò è particolarmente utile per i casi d'uso di intelligenza artificiale (IA), machine learning (ML) e calcolo ad alte prestazioni (HPC).

Il diagramma seguente mostra un pod multi-homed in esecuzione su un nodo worker con più schede di interfaccia di rete (NIC) in uso.




Contesto

Su Amazon EC2, un'interfaccia di rete elastica è un componente di rete logico in un VPC che rappresenta una scheda di rete virtuale. Per molti tipi di istanze EC2, le interfacce di rete condividono una singola scheda di interfaccia di rete (NIC) nell'hardware. Questa singola NIC ha una larghezza di banda e una velocità massime di pacchetti al secondo.

Se la funzionalità Multi-NIC è abilitata, CNI di VPC non assegna gli indirizzi IP in blocco, come avviene per impostazione predefinita. Invece, CNI di VPC assegna un indirizzo IP a un'interfaccia di rete su ciascuna scheda di rete su richiesta all'avvio di un nuovo pod. Tale comportamento riduce la velocità di esaurimento degli indirizzi IP, che è aumentata usando pod multi-homed. Poiché la CNI di VPC assegna l'indirizzo IP su richiesta, l'avvio dei pod potrebbe richiedere più tempo sulle istanze con la funzionalità multi-NIC abilitata.

Considerazioni

- Assicurati che sul tuo cluster Kubernetes sia in esecuzione la versione `1.20.0` CNI di VPC e successive. La funzionalità multi-NIC è disponibile solo nella versione `1.20.0` o successiva di CNI di VPC.
- Abilita la variabile di ambiente `ENABLE_MULTI_NIC` nel plugin CNI di VPC. È possibile eseguire il seguente comando per impostare la variabile e avviare una distribuzione di. DaemonSet
  - `kubectl set env daemonset aws-node -n kube-system ENABLE_MULTI_NIC=true`
- Assicurati di creare nodi worker con più schede di interfaccia di rete (NIC). Per un elenco di istanze EC2 con più schede di interfaccia di rete, consulta Schede di rete nella Guida per l'utente di Amazon EC2.
- Se la funzionalità Multi-NIC è abilitata, CNI di VPC non assegna gli indirizzi IP in blocco, come avviene per impostazione predefinita. Poiché la CNI di VPC assegna l'indirizzo IP su richiesta, l'avvio dei pod potrebbe richiedere più tempo sulle istanze con la funzionalità multi-NIC abilitata. Per ulteriori informazioni, consulta la sezione the section called "Contesto" precedente.
- Con la funzionalità multi-NIC abilitata, per impostazione predefinita i pod non dispongono di più interfacce di rete. È necessario configurare ciascun carico di lavoro per utilizzare multi-NIC. Aggiungi l'annotazione `k8s.amazonaws.com/nicConfig: multi-nic-attachment` ai carichi di lavoro che dovrebbero avere più interfacce di rete.

### Considerazioni sull'IPv6

- Policy IAM personalizzata: per i cluster `IPv6`, crea e utilizza la seguente policy IAM personalizzata per CNI di VPC. Questa policy è specifica per multi-NIC. Per informazioni più generali sull'utilizzo di CNI di VPC con i cluster `IPv6`, consulta the section called "IPv6".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSCNIPolicyIPv6MultiNIC",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:AssignIpv6Addresses",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
```

```
                "ec2:DescribeTags",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstanceTypes",
                "ec2:UnassignIpv6Addresses",
                "ec2:AttachNetworkInterface",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonEKSCNIPolicyENITagIPv6MultiNIC",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        }
    ]
}
```

- `IPv6` Meccanismo di transizione non disponibile: se si utilizza la funzionalità multi-NIC, CNI di VPC non assegna un indirizzo `IPv4` ai pod di un cluster `IPv6`. Altrimenti, CNI di VPC assegna un indirizzo `IPv4` host locale a ciascun pod in modo che un pod possa comunicare con risorse esterne `IPv4` in un altro Amazon VPC o su Internet.

Utilizzo

Dopo aver abilitato la funzionalità multi-NIC nel CNI di VPC e riavviato i pod `aws-node`, è possibile configurare ciascun carico di lavoro in modo che sia multi-homed. Di seguito è riportato un esempio di configurazione YAML con l'annotazione richiesta:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders-deployment
  namespace: ecommerce
  labels:
    app: orders
spec:
  replicas: 3
  selector:
    matchLabels:
      app: orders
```

```
  template:
    metadata:
      annotations:
         k8s.amazonaws.com/nicConfig: multi-nic-attachment
      labels:
        app: orders
    spec:
...
```

Domande frequenti

1. Cos'è una scheda di interfaccia di rete (NIC)?

Una scheda di interfaccia di rete (NIC), chiamata anche semplicemente scheda di rete, è un dispositivo fisico che consente la connettività di rete per l'hardware di cloud computing sottostante. Nei moderni server EC2, si riferisce alla scheda di rete Nitro. Un'interfaccia di rete elastica (ENI) è una rappresentazione virtuale di questa scheda di rete sottostante.

Alcuni tipi di istanze EC2 dispongono di più NIC per prestazioni superiori in termini di larghezza di banda e velocità di pacchetti. In questi casi, è possibile assegnare ENI secondarie alle schede di rete aggiuntive. Ad esempio, ENI #1 può funzionare come interfaccia per la NIC collegata all'indice 0 della scheda di rete, mentre ENI #2 può funzionare come interfaccia per la NIC collegata a un indice di schede di rete separato.

2. Cos'è un pod multi-homed?

Un pod multi-homed è un singolo pod Kubernetes con più interfacce di rete (e per implicazione più indirizzi IP). Ciascun interfaccia di rete di pod è associata a un'interfaccia di rete elastica (ENI) e tali ENI sono rappresentazioni logiche di NIC separate sul nodo worker sottostante. Con più interfacce di rete, un pod multi-homed ha una capacità di trasferimento dati aggiuntiva, che aumenta anche la velocità di trasferimento dei dati.

> **Important**
>
> CNI di VPC può configurare pod multi-homed solo su tipi di istanze con più NIC.

3. Qual è il vantaggio di utilizzare questa funzionalità?

Se hai bisogno di scalare le prestazioni di rete nei tuoi Kubernetes-based carichi di lavoro, puoi utilizzare la funzionalità Multi-NIC per eseguire pod multi-homed che si interfacciano con tutte le NIC

sottostanti a cui è collegato un dispositivo ENA. L'utilizzo di schede di rete aggiuntive aumenta la capacità di larghezza di banda e le prestazioni di velocità di pacchetti nelle applicazioni distribuendo il traffico delle applicazioni su più connessioni simultanee. Ciò è particolarmente utile per i casi d'uso di intelligenza artificiale (IA), machine learning (ML) e calcolo ad alte prestazioni (HPC).

4. Come posso utilizzare questa funzionalità?

1. Innanzitutto, è necessario assicurarsi che il proprio cluster Kubernetes utilizzi versione 1.20 o successiva di CNI di VPC. Per i passaggi per aggiornare CNI di VPC come componente aggiuntivo EKS, consulta the section called "Aggiornamento (componente aggiuntivo EKS)".
2. Quindi, è necessario abilitare il supporto multi-NIC nel CNI di VPC usando la variabile di ambiente `ENABLE_MULTI_NIC`.
3. Quindi, è necessario assicurarsi di creare e unire nodi con più schede di rete. Per un elenco dei tipi di istanze EC2 con più schede di rete, consulta Schede di rete nella Guida per l'utente di Amazon EC2.
4. Infine, configura ciascun carico di lavoro per utilizzare più interfacce di rete (pod multi-homed) o utilizza un'unica interfaccia di rete.

5. Come posso configurare i miei carichi di lavoro per utilizzare più NIC su un nodo worker supportato?

Per utilizzare i pod multi-homed, è necessario aggiungere la seguente annotazione: `k8s.amazonaws.com/nicConfig: multi-nic-attachment`. Ciò collegherà un'ENI da ciascuna NIC nell'istanza sottostante al pod (mappatura da uno a più tra un pod e le NIC).

Se manca questa annotazione, CNI di VPC presuppone che il pod richieda solo 1 interfaccia di rete e gli assegna un IP da un'ENI su qualsiasi NIC disponibile.

6. Quali adattatori di interfaccia di rete sono supportati da questa funzionalità?

È possibile utilizzare qualsiasi adattatore di interfaccia di rete se si dispone di almeno un ENA collegato alla scheda di rete sottostante per il traffico IP. Per ulteriori informazioni, consulta Adattatore elastico di rete (ENA) nella Guida per l'utente di Amazon EC2.

Configurazioni dei dispositivi di rete supportate:

- Le interfacce ENA offrono tutte le tradizionali funzionalità di rete e routing IP necessarie per supportare la rete IP per un VPC. Per ulteriori informazioni, consulta Abilitazione delle reti avanzate con ENI su istanze EC2.

- Le interfacce EFA (EFA con ENA) offrono sia il dispositivo ENA per le reti IP che il dispositivo EFA per le comunicazioni a bassa latenza e a throughput elevato.

> **Important**
>
> Se a una scheda di rete è collegato solo un EFA-only adattatore, il VPC CNI lo ignorerà durante il provisioning della connettività di rete per un pod multi-homed. Tuttavia, se si combina un EFA-only adattatore con un adattatore ENA su una scheda di rete, il VPC CNI gestirà gli ENI anche su questo dispositivo. Per utilizzare le EFA-only interfacce con i cluster EKS, vedere. the section called "Configurazione dei cluster di formazione con EFA"

7. Posso vedere se un nodo del mio cluster supporta ENA?

Sì, puoi utilizzare la AWS CLI o l'API EC2 per recuperare informazioni di rete su un'istanza EC2 nel tuo cluster. Ciò fornisce dettagli sul fatto che l'istanza disponga o meno del supporto ENA. Nell'esempio seguente, sostituisci `<your-instance-id>` con l'ID di istanza EC2 di un nodo.

AWS Esempio di CLI:

```
aws ec2 describe-instances --instance-ids <your-instance-id> --query
 "Reservations[].Instances[].EnaSupport"
```

Output di esempio:

```
[ true ]
```

8. Posso vedere i diversi indirizzi IP associati a un pod?

No, non è semplice. Tuttavia, è possibile utilizzare `nsenter` dal nodo per eseguire strumenti di rete comuni come `ip route show` e visualizzare gli indirizzi IP e le interfacce aggiuntivi.

9. Posso controllare il numero di interfacce di rete per i miei pod?

No. Quando il carico di lavoro è configurato per utilizzare più NIC su un'istanza supportata, un singolo pod ha automaticamente un indirizzo IP da ciascuna scheda di rete sull'istanza. In alternativa, i pod single-homed avranno un'interfaccia di rete collegata a una NIC sull'istanza.

> **Important**
>
> Le schede di rete a cui è collegato solo un EFA-only dispositivo vengono ignorate dal VPC CNI.

10. Posso configurare i miei pod per utilizzare una scheda NIC specifica?

No, questa funzionalità non è supportata. Se un pod ha l'annotazione pertinente, CNI di VPC lo configura automaticamente per utilizzare ogni NIC con un adattatore ENA sul nodo worker.

11. Questa funzionalità funziona con le altre funzionalità di rete di CNI di VPC?

Sì, la funzionalità multi-NIC di CNI di VPC funziona sia con reti personalizzate che con la rilevazione avanzata delle sottoreti. Tuttavia, i pod multi-homed non utilizzano sottoreti o gruppi di sicurezza personalizzati. Invece, CNI di VPC assegna indirizzi IP e interfacce di rete ai pod multi-homed con la stessa configurazione di sottorete e gruppo di sicurezza del nodo. Per ulteriori informazioni sulle reti personalizzate, consulta the section called "Rete personalizzata".

La funzionalità multi-NIC di CNI di VPC non funziona e non può essere combinata con i gruppi di sicurezza per i pod.

12. Posso utilizzare le politiche di rete con questa funzionalità?

Sì, è possibile utilizzare le policy di rete Kubernetes con multi-NIC. Le policy di rete Kubernetes limitano il traffico di rete verso e dai pod. Per ulteriori informazioni sull'applicazione delle policy di rete con CNI di VPC, consulta the section called "Policy di Kubernetes".

13. Il supporto multi-NIC è abilitato in modalità automatica EKS?

Multi-NIC non è supportato per i cluster EKS Auto Mode.

## Plug-in CNI alternativi per cluster Amazon EKS

Il plug-in CNI di Amazon VPC per Kubernetes è l'unico plug-in supportato da Amazon EKS con nodi Amazon EC2. Amazon EKS supporta le funzionalità principali di Cilium e Calico per Amazon EKS Hybrid Nodes. Amazon EKS esegue Kubernetes a monte, permettendo di installare plug-in CNI compatibili alternativi sui nodi Amazon EC2 del cluster. Se hai nodi Fargate nel cluster, il plug-in CNI di Amazon VPC per Kubernetes è già presente sui nodi Fargate. È l'unico plug-in CNI che puoi utilizzare con i nodi Fargate. Qualsiasi tentativo di installare un plug-in CNI alternativo sui nodi Fargate avrà esito negativo.

Se prevedi di utilizzare un plug-in CNI alternativo sui nodi Amazon EC2, ti consigliamo vivamente di ottenere supporto commerciale o di disporre delle competenze interne per risolvere i problemi e contribuire alle correzioni del progetto del plug-in CNI.

Amazon EKS dispone di una rete di partner che offrono supporto per plug-in CNI compatibili alternativi. Per informazioni dettagliate sulle versioni, sulle qualifiche e sui test eseguiti, consulta la documentazione dei partner riportata di seguito.

| Partner | Prodotto | Documentazione |
|---|---|---|
| Tigera | Calico | Istruzioni di installazione |
| Isovalent | Cilium | Istruzioni di installazione |
| Juniper | Cloud-Native Rete Contrail (CN2) | Istruzioni di installazione |
| VMware | Antrea | Istruzioni di installazione |

Amazon EKS mira a dare all'utente una vasta gamma di opzioni per coprire tutti i casi d'uso.

## Plug-in alternativi compatibili per le policy di rete

Calico è una soluzione ampiamente adottata per il networking e la sicurezza dei container. L'utilizzo di Calico su EKS offre un'applicazione delle policy di rete completamente conforme per i cluster EKS. Inoltre, puoi scegliere di utilizzare la rete di Calico, che conserva gli indirizzi IP dal tuo VPC sottostante. Calico Cloud migliora le funzionalità di Calico Open Source, fornendo funzionalità avanzate di sicurezza e osservabilità.

Il flusso di traffico da e verso i pod con gruppi di sicurezza associati non è soggetto all'applicazione di policy di rete ed è limitato esclusivamente all'applicazione dei gruppi di sicurezza Amazon VPC.

Se utilizzi l'applicazione della policy di rete di Calico, consigliamo di impostare la variabile di ambiente `ANNOTATE_POD_IP` su `true` per evitare un problema noto con Kubernetes. Per utilizzare questa funzione, è necessario aggiungere l'`patch`autorizzazione per i pod a. `aws-node` ClusterRole Tieni presente che l'aggiunta di autorizzazioni per le patch `aws-node` DaemonSet aumenta l'ambito di sicurezza del plug-in. Per ulteriori informazioni, consulta ANNOTATE_POD_IP nel repository VPC CNI attivo. GitHub

### Considerazioni per la modalità automatica di Amazon EKS

La modalità automatica di Amazon EKS non supporta plug-in CNI o plug-in di policy di rete alternativi. Per ulteriori informazioni, consulta Modalità automatica EKS.

## Collegare più interfacce di rete ai pod con Multus

Multus CNI è un plug-in CNI (Container Network Interface) per Amazon EKS che consente di allegare più interfacce di rete a un pod. Per ulteriori informazioni, consulta la Multus-CNI documentazione su GitHub.

In Amazon EKS, ogni pod ha un'interfaccia di rete assegnata dal plug-in CNI di Amazon VPC. Con Multus, è possibile creare un pod multi-homed con più interfacce. Ciò viene ottenuto da Multus che agisce come un «meta-plugin»; un plugin CNI che può richiamare diversi altri plugin CNI. AWS il supporto per Multus viene configurato con il plug-in Amazon VPC CNI come plug-in predefinito per i delegati.

- Amazon EKS non creerà e pubblicherà plugin CNI Single Root I/O Virtualization (SR-IOV) e Data Plane Development Kit (DPDK). Tuttavia, è possibile ottenere l'accelerazione dei pacchetti collegandosi direttamente ad Amazon EC2 Elastic Network Adapters (ENA) tramite il dispositivo host gestito Multus e plug-in `ipvlan`.
- Amazon EKS supporta Multus, che fornisce un processo generico che consente un semplice concatenamento di plug-in CNI aggiuntivi. Multus e il processo di concatenamento sono supportati, ma AWS non forniranno supporto per tutti i plug-in CNI compatibili che possono essere concatenati o per i problemi che potrebbero insorgere in quei plug-in CNI non correlati alla configurazione di concatenamento.
- Amazon EKS fornisce supporto e gestione del ciclo di vita per il plug-in Multus, ma non è responsabile di alcun indirizzo IP o gestione aggiuntiva associata alle interfacce di rete aggiuntive. L'indirizzo IP e la gestione dell'interfaccia di rete predefinita che utilizza il plug-in CNI di Amazon VPC rimangono invariati.
- Solo il plug-in CNI di Amazon VPC è ufficialmente supportato come plug-in delegato predefinito. Se si sceglie di non utilizzare il plug-in CNI di Amazon VPC per la rete primaria, è necessario modificare il manifesto di installazione Multus pubblicato per riconfigurare il plug-in delegato predefinito in un CNI alternativo.
- Multus è supportato solo quando si utilizza il CNI di Amazon VPC come CNI principale. Non supportiamo il CNI di Amazon VPC se utilizzato per interfacce di ordine superiore, secondarie o di altro tipo.

- Per evitare che il plug-in CNI di Amazon VPC tenti di gestire interfacce di rete aggiuntive assegnate ai pod, aggiungi il seguente tag alle interfacce di rete:

  key

  : `node.k8s.amazonaws.com/no_manage`

  value

  : `true`

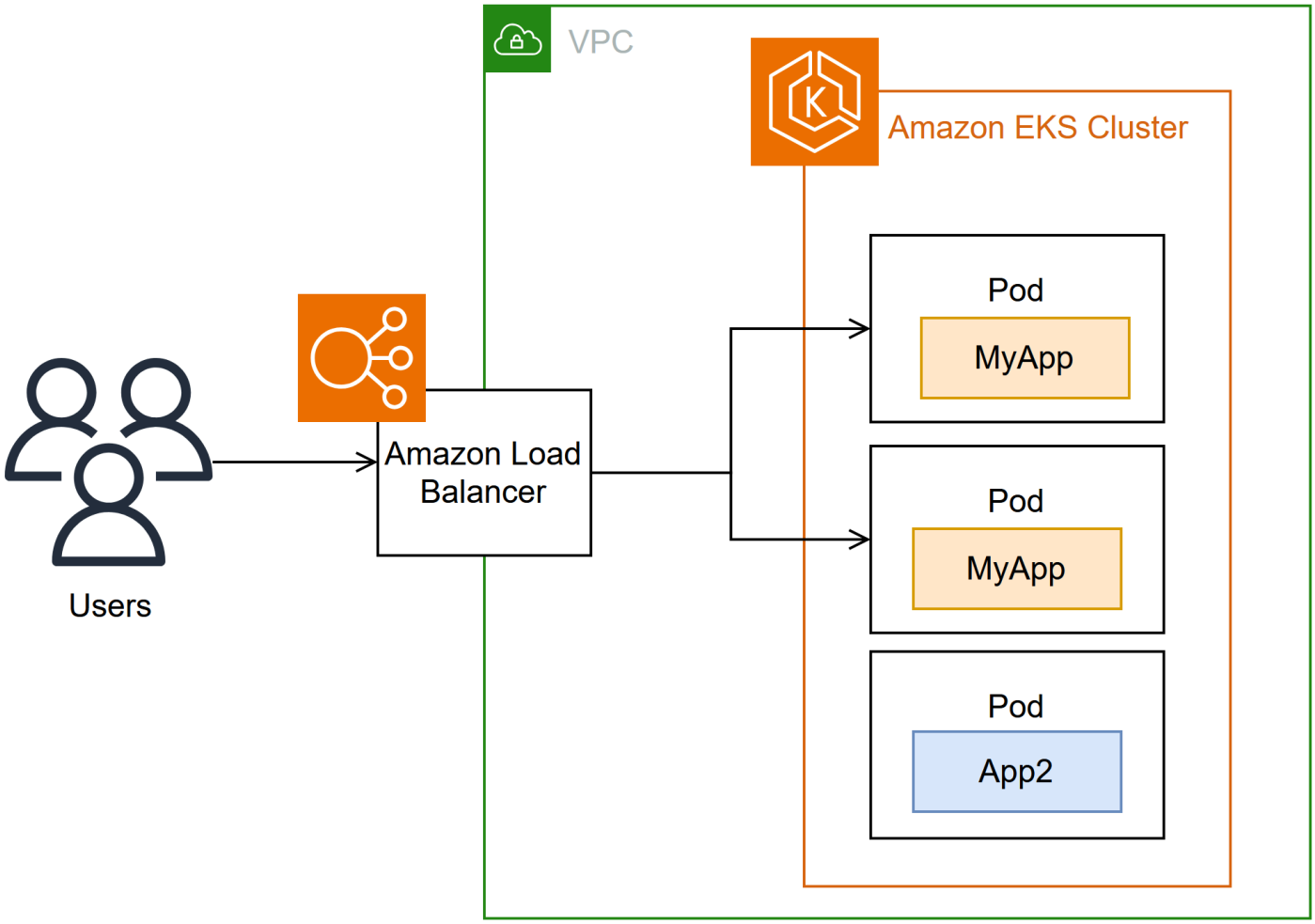
- Multus è compatibile con policy di rete, ma queste devono essere arricchite per includere porte e indirizzi IP che possono far parte di interfacce di rete aggiuntive allegate ai pod.

Per una guida dettagliata all'implementazione, consulta la Multus Setup Guide su. https://github.com/aws-samples/eks-install-guide-for-multus/blob/main/README.md GitHub

## Indirizza il traffico Internet con AWS Controller del sistema di bilanciamento del carico

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Il AWS Load Balancer Controller gestisce AWS Elastic Load Balancer per un cluster Kubernetes. Puoi usare il controller per esporre le app del cluster a Internet. Il controller fornisce sistemi di AWS bilanciamento del carico che puntano alle risorse Cluster Service o Ingress. In altri termini, il controller crea un unico indirizzo IP o nome DNS che punta a più pod del cluster.




Il controller monitora le risorse di Kubernetes Ingress o Service. In risposta, crea le risorse AWS Elastic Load Balancing appropriate. Puoi configurare il comportamento specifico dei bilanciatori del carico applicando annotazioni alle risorse Kubernetes. Ad esempio, è possibile collegare gruppi di AWS sicurezza ai sistemi di bilanciamento del carico utilizzando le annotazioni.

Il controller fornisce le risorse seguenti:

Kubernetes **`Ingress`**

L'LBC crea un AWS Application Load Balancer (ALB) quando crei un `Ingress`. Rivedi le annotazioni che puoi applicare a una risorsa Ingress.

Servizio Kubernetes di tipo **`LoadBalancer`**

LBC crea un AWS Network Load Balancer (NLB) quando crei un servizio Kubernetes di tipo. `LoadBalancer` Rivedi le annotazioni che puoi applicare a una risorsa Service.

In passato, il Network Load Balancer (NLB) Kubernetes veniva utilizzato per le destinazioni istanza, ma per le destinazioni IP veniva utilizzato l'LBC. Con la versione AWS Load Balancer Controller `2.3.0` o successiva, puoi creare NLB utilizzando entrambi i tipi di target. Per ulteriori informazioni sui tipi di destinazioni dei NLB, consultare Target type (Tipo di destinazione) nella Guida per l'utente di Network Load Balancer.

Il controller è un progetto open source gestito su. GitHub

Prima di implementare il controller, è consigliabile esaminare i prerequisiti e le considerazioni in Route application and HTTP traffic with Application Load Balancers e the section called "Bilanciamento del carico di rete". In questi argomenti, verrà distribuita un'app di esempio che include un sistema di AWS bilanciamento del carico.

API Kubernetes **`Gateway`**

Con la versione AWS Load Balancer Controller `2.14.0` o successiva, LBC crea un Application Load AWS Balancer (ALB) quando crei un Kubernetes. `Gateway` Kubernetes Gateway standardizza più configurazioni rispetto a Ingress, che richiedeva annotazioni personalizzate per molte opzioni comuni. Rivedi la configurazione che puoi applicare a una risorsa Gateway. Per ulteriori informazioni sull'`Gateway`API, consulta Gateway API nella documentazione di Kubernetes.

## Installare il controller

È possibile utilizzare una delle seguenti procedure per installare il AWS Load Balancer Controller:

- Se non conosci Amazon EKS, ti consigliamo di utilizzare Helm per l'installazione perché semplifica l'installazione del Load AWS Balancer Controller. Per ulteriori informazioni, consulta the section called "Installa con Helm".
- Per configurazioni avanzate, come cluster con accesso di rete limitato ai registri di container pubblici, utilizza i manifesti Kubernetes. Per ulteriori informazioni, consulta the section called "Installa con i manifesti".

## Migrazione dalle versioni obsolete del controller

- Se sono installate versioni obsolete del Load AWS Balancer Controller, consulta. the section called "Esegui la migrazione da obsoleto"

- Le versioni obsolete non possono essere aggiornate. Devono essere rimossi e deve essere installata una versione corrente del AWS Load Balancer Controller.
- Le versioni obsolete includono:
  - AWS ALB Ingress Controller for Kubernetes («Ingress Controller»), un predecessore del Load Balancer Controller. AWS
  - Qualsiasi `0.1.x` versione del AWS Load Balancer Controller

## Provider cloud legacy

Kubernetes include un provider cloud legacy per AWS. Il provider cloud legacy è in grado di effettuare il provisioning di sistemi di bilanciamento del AWS carico, in modo simile al Load Balancer Controller AWS . Il provider cloud legacy crea i Classic Load Balancer. Se non installi il AWS Load Balancer Controller, Kubernetes utilizzerà per impostazione predefinita il provider cloud legacy. È necessario installare il AWS Load Balancer Controller ed evitare di utilizzare il provider cloud legacy.

**Important**

Nelle versioni 2.5 e successive, il AWS Load Balancer Controller diventa il controller predefinito per le risorse del servizio Kubernetes e crea un Network Load AWS Balancer (NLB) per ogni servizio. `type: LoadBalancer` Lo fa creando un webhook mutante per i servizi, che imposta il campo `spec.loadBalancerClass` a `service.k8s.aws/nlb` per nuovi servizi di `type: LoadBalancer`. È possibile disattivare questa funzione e tornare a utilizzare il Cloud Provider obsoleto come controller predefinito, impostando il valore della tabella di comando `enableServiceMutatorWebhook` a `false`. Il cluster non eseguirà il provisioning dei nuovi Classic Load Balancer per i tuoi servizi a meno che non disattivi questa funzionalità. I sistemi Classic Load Balancer esistenti continueranno a funzionare.

## Installa AWS Controller Load Balancer con timone

**Tip**

Registrati ai prossimi workshop Amazon EKS.

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non devi installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

Questo argomento descrive come installare il AWS Load Balancer Controller utilizzando Helm, un gestore di pacchetti per Kubernetes, e. `eksctl` Il controller viene installato con le opzioni predefinite. Per ulteriori informazioni sul controller, inclusi i dettagli sulla configurazione con annotazioni, vedere la documentazione del controller Load AWS Balancer su. GitHub

Nei passaggi seguenti, sostituisci i valori di esempio con i tuoi valori.

Prerequisiti

Prima di iniziare questo tutorial, completa i seguenti passaggi:

- Crea un cluster Amazon EKS. Per crearne uno, consulta Nozioni di base.
- Installa i Gateway API CRD sul cluster Amazon EKS.
- Installa Helm sul computer locale.
- Assicurati che il plug-in CNI di Amazon VPC per Kubernetes, `kube-proxy` e i componenti aggiuntivi CoreDNS siano alle versioni minime elencate in Service account tokens.
- Scopri i concetti di AWS Elastic Load Balancing. Per ulteriori informazioni, consulta la Guida per l'utente di Elastic Load Balancing.
- Scopri il servizio Kubernetes e le risorse in entrata.

Considerazioni

Prima di procedere con i passaggi per la configurazione descritti in questa pagina, considera quanto segue:

- La policy e il ruolo IAM (`AmazonEKSLoadBalancerControllerRole`) possono essere riutilizzati su più cluster EKS nello stesso account. AWS
- Se stai installando il controller sullo stesso cluster in cui il ruolo (`AmazonEKSLoadBalancerControllerRole`) è stato originariamente creato, vai a Step 2: Install Load Balancer Controller dopo aver verificato l'esistenza del ruolo.

- Se utilizzato, IRSA (IAM Roles for Service Accounts) deve essere configurato per ogni cluster e l'ARN del provider OpenID Connect (OIDC) nella policy di attendibilità del ruolo è specifico per ogni cluster EKS. Inoltre, se stai installando il controller su un nuovo cluster con un`AmazonEKSLoadBalancerControllerRole` esistente, aggiorna la policy di attendibilità del ruolo per includere il provider OIDC del nuovo cluster e crea un nuovo account di servizio con l'annotazione di ruolo appropriata. Per determinare se disponi già di un provider OIDC o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

### `Fase 1: Creare un ruolo IAM utilizzando eksctl`

I passaggi seguenti si riferiscono alla versione di rilascio del AWS Load Balancer Controller v2.14.1. Per ulteriori informazioni su tutte le versioni, consulta la pagina di rilascio del AWS Load Balancer Controller su. GitHub

1. Scarica una policy IAM per il AWS Load Balancer Controller che gli consenta di effettuare chiamate alle AWS API per tuo conto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy.json
```

   - Se ti trovi in una AWS partizione non standard, ad esempio un'area governativa o cinese, consulta le politiche GitHub e scarica la politica appropriata per la tua regione.

2. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> Se visualizzi la politica in Console di gestione AWS, la console mostra gli avvisi per il servizio ELB, ma non per il servizio ELB v2. Ciò accade perché alcune azioni nella policy esistono per ELB v2, ma non per ELB. Queste avvertenze per ELB possono essere ignorate.

3. Sostituisci i valori per il nome del cluster, il codice regionale e l'ID dell'account.

```
eksctl create iamserviceaccount \
```

```
    --cluster=<cluster-name> \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws: iam::<AWS_ACCOUNT_ID>:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region <aws-region-code> \
    --approve
```

Fase 2: Installazione AWS Controller del sistema di bilanciamento del carico

1. Aggiungi l'archivio `eks-charts` cartografico Helm. AWS mantiene questo repository attivo. GitHub

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Aggiornare il repository locale per assicurarsi di avere i grafici più recenti.

```
helm repo update eks
```

3. Installa il AWS Load Balancer Controller.

   Se stai implementando il controller su nodi Amazon EC2 che hanno accesso limitato al servizio di metadati di istanza (IMDS) di Amazon EC2 o se stai implementando in Fargate o Amazon EKS Hybrid Nodes, aggiungi questi flag al comando `helm` seguente:

   - `--set region=`*`region-code`*
   - `--set vpcId=`*`vpc-xxxxxxxx`*

   Sostituisci *`my-cluster`* con il nome del cluster. Nel comando seguente, `aws-load-balancer-controller` indica l'account del servizio Kubernetes creato in una fase precedente.

   Per ulteriori informazioni sulla configurazione del grafico di comando, vedere values.yaml su. GitHub

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
```

```
  --version 1.14.0
```

> **Important**
>
> Il grafico implementato non riceve automaticamente gli aggiornamenti di sicurezza. È necessario eseguire manualmente l'aggiornamento a un grafico più recente quando diventa disponibile. Durante l'aggiornamento, passate al comando precedente. *`install`* `upgrade`

Il comando `helm install` installa automaticamente le definizioni di risorse personalizzate (CRD) per il controller. Il comando `helm upgrade` non lo fa. Se utilizzi `helm upgrade`,, devi installare manualmente le CRD. Esegui il comando seguente per installare le CRD.

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-
controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Passaggio 3: verifica che il controller sia installato

1. Verificare che il controller sia installato.

   ```
   kubectl get deployment -n kube-system aws-load-balancer-controller
   ```

   Di seguito viene riportato un output di esempio:

   ```
   NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
   aws-load-balancer-controller   2/2     2            2           84s
   ```

   Riceverai l'output precedente se hai eseguito l'implementazione con Helm. Se hai eseguito l'implementazione utilizzando il manifesto Kubernetes, hai una sola replica.

2. Prima di utilizzare il controller per il provisioning AWS delle risorse, il cluster deve soddisfare requisiti specifici. Per ulteriori informazioni, consultare the section called "Bilanciamento del carico di applicazione" e the section called "Bilanciamento del carico di rete".

# Installa AWS Load Balancer Controller con manifesti

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non è necessario installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

Questo argomento descrive come installare il controller scaricando e applicando i manifesti di Kubernetes. É possibile visualizzare l'intera documentazione per il controller su GitHub.

Nei passaggi seguenti, sostituisci i valori di esempio con i tuoi valori.

## Prerequisiti

Prima di iniziare questo tutorial, completa i seguenti passaggi:

- Crea un cluster Amazon EKS. Per crearne uno, consulta Nozioni di base.
- Installa Helm sul computer locale.
- Assicurati che il plug-in CNI di Amazon VPC per Kubernetes, `kube-proxy` e i componenti aggiuntivi CoreDNS siano alle versioni minime elencate in Service account tokens.
- Scopri i concetti di AWS Elastic Load Balancing. Per ulteriori informazioni, consulta la Guida per l'utente di Elastic Load Balancing.
- Scopri il servizio Kubernetes e le risorse in entrata.

## Considerazioni

Prima di procedere con i passaggi per la configurazione descritti in questa pagina, considera quanto segue:

- La policy e il ruolo IAM (`AmazonEKSLoadBalancerControllerRole`) possono essere riutilizzati su più cluster EKS nello stesso account. AWS
- Se stai installando il controller sullo stesso cluster in cui il ruolo (`AmazonEKSLoadBalancerControllerRole`) è stato originariamente creato, vai a Step 2: Install cert-manager dopo aver verificato l'esistenza del ruolo.

- Se utilizzato, IRSA (IAM Roles for Service Accounts) deve essere configurato per ogni cluster e l'ARN del provider OpenID Connect (OIDC) nella policy di attendibilità del ruolo è specifico per ogni cluster EKS. Inoltre, se stai installando il controller su un nuovo cluster con un`AmazonEKSLoadBalancerControllerRole` esistente, aggiorna la policy di attendibilità del ruolo per includere il provider OIDC del nuovo cluster e crea un nuovo account di servizio con l'annotazione di ruolo appropriata. Per determinare se disponi già di un provider OIDC o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Passaggio 1: configura IAM

I passaggi seguenti si riferiscono alla versione di rilascio del AWS Load Balancer Controller v2.14.1. Per ulteriori informazioni su tutte le versioni, consulta la pagina di rilascio del AWS Load Balancer Controller su. GitHub

1. Scarica una policy IAM per il AWS Load Balancer Controller che gli consenta di effettuare chiamate per tuo AWS APIs conto.

   Example

   AWS

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/install/iam_policy_us-gov.json
   ```

   ```
   mv iam_policy_us-gov.json iam_policy.json
   ```

2. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

   ```
   aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy.json
   ```

> **Note**
>
> Se visualizzi la policy in Console di gestione AWS, la console mostra gli avvisi per il servizio ELB, ma non per il servizio ELB v2. Ciò accade perché alcune azioni nella policy esistono per ELB v2, ma non per ELB. Queste avvertenze per ELB possono essere ignorate.

Example

eksctl

a. Sostituisci *my-cluster* con il nome del cluster e *111122223333* con il tuo ID account, quindi esegui il comando.

```
eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

a. Recupera l'ID del provider OIDC del cluster e memorizzalo in una variabile.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

b. Determina se un provider OIDC IAM con l'ID del cluster è già presente nell'account. È necessario configurare OIDC sia per il cluster che per IAM.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se viene restituito un output, devi creare un provider OIDC IAM per il tuo cluster. Se non viene restituito alcun output, devi creare un provider OIDC IAM per il cluster. Per ulteriori informazioni, consulta the section called "Gestore di identità OIDC IAM".

c. Copia i seguenti contenuti sul dispositivo. Sostituisci *111122223333* con l'ID del tuo account. *region-code*Sostituiscilo con la AWS regione in cui si trova il cluster. Sostituisci *EXAMPLED539D4633E53DE1B71EXAMPLE* con l'output restituito nella fase precedente.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
                }
            }
        }
    ]
}
```

d. Creare il ruolo IAM.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

e. Allegare la policy IAM gestita da Amazon EKS richiesta al ruolo IAM. Sostituisci *111122223333* con l'ID del tuo account.

```
aws iam attach-role-policy \
```

```
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

f. Copia i seguenti contenuti sul dispositivo. Sostituisci *111122223333* con l'ID del tuo account. Dopo aver sostituito il testo, esegui il comando modificato per creare il file `aws-load-balancer-controller-service-account.yaml`.

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKSLoadBalancerControllerRole
EOF
```

g. Creare l'account del servizio Kubernetes nel cluster. L'account del servizio Kubernetes denominato `aws-load-balancer-controller` è annotato con il ruolo IAM che hai creato denominato. *AmazonEKSLoadBalancerControllerRole*

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

Passaggio 2: installa **`cert-manager`**

Installa `cert-manager` utilizzando uno dei metodi seguenti per inserire la configurazione del certificato nei Webhook. Per ulteriori informazioni, consulta Getting Started nella documentazione cert-manager.

Si consiglia di utilizzare il registro dei container `quay.io` per installare `cert-manager`. Se i tuoi nodi non hanno accesso al registro dei container `quay.io`, installa `cert-manager` utilizzando Amazon ECR (vedi sotto).

Example

Quay.io

a. Se i tuoi nodi hanno accesso al registro del container `quay.io`, installa `cert-manager` per inserire la configurazione del certificato nei Webhook.

```
kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

Amazon ECR

a. Installa `cert-manager` utilizzando uno dei metodi seguenti per inserire la configurazione del certificato nei Webhook. Per ulteriori informazioni, consulta Getting Started nella documentazione cert-manager.
b. Eseguire il download del manifesto.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

c. Estrai le seguenti immagini e inviale a un repository a cui i tuoi nodi hanno accesso. Per ulteriori informazioni su come estrarre, taggare e inviare immagini al tuo repository, consulta la sezione the section called "Copia di un'immagine in un repository".

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

d. Nel manifesto, sostituisci `quay.io` per le tre immagini con il nome del tuo registro. Il comando seguente presuppone che il nome del repository privato sia lo stesso del repository di origine. Sostituisci *111122223333.dkr.ecr.region-code.amazonaws.com* con il tuo registro privato.

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./cert-manager.yaml
```

e. Applicare il file manifesto.

```
kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

Fase 3: Installare AWS Load Balancer Controller

1. Scaricare le specifiche del controller. Per ulteriori informazioni sull'utilizzo del controller, consulta la documentazione su GitHub.

```
curl -Lo v2_14_1_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.14.1/v2_14_1_full.yaml
```

2. Apporta le seguenti modifiche al file.

   a. Se hai scaricato il file `v2_14_1_full.yaml`, emetti il seguente comando per rimuovere la sezione `ServiceAccount` nel manifesto. Se non rimuovi questa sezione, l'annotazione richiesta apportata all'account del servizio in un passaggio precedente viene sovrascritta. Inoltre, se elimini il controller, conserva l'account del servizio creato in una fase precedente.

   ```
   sed -i.bak -e '764,772d' ./v2_14_1_full.yaml
   ```

   Se hai scaricato una versione diversa del file, apri il file in un editor e rimuovi le righe seguenti.

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     labels:
       app.kubernetes.io/component: controller
       app.kubernetes.io/name: aws-load-balancer-controller
     name: aws-load-balancer-controller
     namespace: kube-system
   ---
   ```

   b. Sostituisci `your-cluster-name` nella sezione `Deployment spec` del file con il nome del cluster sostituendo *`my-cluster`* con il nome del tuo cluster.

   ```
   sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_14_1_full.yaml
   ```

c. Se i tuoi nodi non hanno accesso ai repository di immagini Amazon EKS Amazon ECR, devi estrarre la seguente immagine e inviarla a un repository a cui i tuoi nodi hanno accesso. Per ulteriori informazioni su come estrarre, taggare e inviare un'immagine al tuo repository, consulta la sezione the section called "Copia di un'immagine in un repository".

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.14.1
```

Aggiungi il nome del tuo registro al manifesto. Il comando seguente presuppone che il nome del repository privato sia lo stesso del repository di origine e aggiunge il nome del tuo registro privato al file. Sostituisci *111122223333.dkr.ecr.region-code.amazonaws.com* con il tuo registro. Questa riga presuppone che il nome del repository privato sia lo stesso del repository di origine. In caso contrario, sostituisci il testo `eks/aws-load-balancer-controller` dopo il nome del registro privato con il nome del tuo repository.

```
sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-controller|
111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-balancer-
controller|' ./v2_14_1_full.yaml
```

d. (Richiesto solo per Fargate o IMDS con restrizioni)

Se stai implementando il controller su nodi Amazon EC2 che hanno accesso limitato al servizio di metadati di istanza (IMDS) di Amazon EC2 o se stai implementando su Fargate o Amazon EKS Hybrid Nodes, aggiungi i `following parameters` in `- args:`.

```
[...]
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. Applicare il file.

```
kubectl apply -f v2_14_1_full.yaml
```

4. Scarica il manifesto di `IngressClass` e `IngressClassParams` sul cluster.

```
curl -Lo v2.14.1_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.14.1/v2_14_1_ingclass.yaml
```

5. Applica il file manifesto al cluster.

```
kubectl apply -f v2.14.1_ingclass.yaml
```

Passaggio 4: verifica che il controller sia installato

1. Verificare che il controller sia installato.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

   Di seguito viene riportato un output di esempio:

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

   Riceverai l'output precedente se hai eseguito l'implementazione con Helm. Se hai eseguito l'implementazione utilizzando il manifesto Kubernetes, hai una sola replica.
2. Prima di utilizzare il controller per il provisioning AWS delle risorse, il cluster deve soddisfare requisiti specifici. Per ulteriori informazioni, consultare the section called "Bilanciamento del carico di applicazione" e the section called "Bilanciamento del carico di rete".

## Esegui la migrazione delle app da Controller di ingresso ALB obsoleto

Questo argomento descrive come migrare da versioni obsolete del Controller. Più specificamente, descrive come rimuovere le versioni obsolete del Load Balancer AWS Controller.

- Le versioni obsolete non possono essere aggiornate. Devi prima rimuoverle e quindi installare una versione attuale.
- Le versioni obsolete includono:
  - AWS ALB Ingress Controller for Kubernetes («Ingress Controller»), un predecessore del Load Balancer Controller. AWS
  - Qualsiasi `0.1.x` versione del AWS Load Balancer Controller

Rimuovi la versione obsoleta del controller

> **Note**
>
> Potresti aver installato la versione obsoleta utilizzando Helm o manualmente con i manifesti di Kubernetes. Completare la procedura utilizzando lo strumento con cui è stata installata in origine.

1. Se il grafico Helm `incubator/aws-alb-ingress-controller` è installato, procedi alla disinstallazione.

   ```
   helm delete aws-alb-ingress-controller -n kube-system
   ```

2. Se la versione `0.1.x` del grafico `eks-charts/aws-load-balancer-controller` è installata, procedi alla disinstallazione. L'aggiornamento dalla versione `0.1.x` alla versione `1.0.0` non ha esito positivo a causa di incompatibilità con la versione dell'API webhook.

   ```
   helm delete aws-load-balancer-controller -n kube-system
   ```

3. Verificare che il controller sia già installato.

   ```
   kubectl get deployment -n kube-system alb-ingress-controller
   ```

   Questo è l'output restituito se il controller non è installato.

   ```
   Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
   ```

   Questo è l'output restituito se il controller è installato.

   ```
   NAME                   READY UP-TO-DATE AVAILABLE AGE
   alb-ingress-controller 1/1   1          1         122d
   ```

4. Immettere i seguenti comandi per rimuovere il controller.

   ```
   kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
   controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
   kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
   controller/v1.1.8/docs/examples/rbac-role.yaml
   ```

Migrazione a AWS Load Balancer Controller

Per migrare dall'ALB Ingress Controller for Kubernetes al Load AWS Balancer Controller, devi:

1. Rimuovere il Controller di ingresso ALB (vedi sopra).
2. Installa il AWS Load Balancer Controller.
3. Aggiungi una policy aggiuntiva al ruolo IAM utilizzato dal AWS Load Balancer Controller. Questa policy consente a LBC la gestione delle risorse create dal Controller di ingresso ALB per Kubernetes.
4. Scaricare la policy IAM. Questa policy consente al AWS Load Balancer Controller di gestire le risorse create dall'ALB Ingress Controller per Kubernetes. É anche possibile consultare la policy.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy_v1_to_v2_additional.json
```

5. Se il cluster si trova nelle regioni AWS GovCloud (Stati Uniti orientali) o AWS GovCloud (Stati Uniti occidentali), sostituiscilo con. AWS `arn:aws:` `arn:aws-us-gov:`

```
sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

6. Creare la policy IAM e annotate l'ARN restituito.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

7. Collega la policy IAM al ruolo IAM utilizzato dal AWS Load Balancer Controller. Sostituiscila *`your-role-name`* con il nome del ruolo, ad esempio`AmazonEKSLoadBalancerControllerRole`.

   Se hai creato il ruolo utilizzando`eksctl`, allora per trovare il nome del ruolo che è stato creato, apri la AWS CloudFormation console e seleziona lo stack eksctl- ***`my-cluster`*** - addon-iamserviceaccount-kube-system - aws-load-balancer-controller. Selezionare la scheda Risorse. Il nome del ruolo è nella colonna ID fisico.

```
aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerAdditionalIAMPolicy
```

# Gestione di CoreDNS per DNS nei cluster Amazon EKS

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non devi installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

CoreDNS è un server DNS flessibile ed estensibile che può fungere da cluster DNS Kubernetes. Quando si avvia un cluster Amazon EKS con almeno un nodo, due repliche dell'immagine CoreDNS vengono distribuite per impostazione predefinita, indipendentemente dal numero di nodi distribuiti nel cluster. I contenitori CoreDNS forniscono la risoluzione dei nomi per tutti i pod del cluster. I pod CoreDNS possono essere distribuiti ai nodi Fargate se il cluster include un profilo Fargate con un namespace che corrisponde al namespace del `deployment` CoreDNS. Per ulteriori informazioni su CoreDNS, consultare Utilizzo di CoreDNS per l'individuazione dei servizi nella documentazione Kubernetes.

## Versioni di CoreDNS

La tabella seguente elenca la versione più recente del componente aggiuntivo di Amazon EKS disponibile per ogni versione di Kubernetes.

| Versione di Kubernetes | Versione di CoreDNS |
|---|---|
| 1,36 | v1.14.3-eksbuild.16 |
| 1,35 | v1.14.3-eksbuild.16 |
| 1,34 | v1.13.2-eksbuild.24 |
| 1.33 | v1.12.4-eksbuild.31 |
| 1,32 | v1.11.4-eksbuild.53 |
| 1,31 | v1.11.4-eksbuild.53 |

> **Important**
>
> Se gestisci autonomamente questo componente aggiuntivo, le versioni nella tabella potrebbero non essere le stesse di quelle autogestite disponibili. Per ulteriori informazioni sull'aggiornamento del componente aggiuntivo del tipo autogestito, consulta the section called "Aggiornamento (autogestito)".

## Considerazioni importanti sull'aggiornamento di CoreDNS

- Gli aggiornamenti CoreDNS utilizzano un per aiutare a mantenere la disponibilità del servizio DNS durante il processo di PodDisruptionBudget aggiornamento.
- Per migliorare la stabilità e la disponibilità di Implementazione di CoreDNS, le versioni `v1.9.3-eksbuild.6` e successive e `v1.10.1-eksbuild.3` vengono distribuite con un `PodDisruptionBudget`. Se hai implementato un `PodDisruptionBudget` esistente, l'aggiornamento a queste versioni potrebbe non riuscire. Se l'aggiornamento non riesce, il problema dovrebbe essere risolto eseguendo una delle seguenti operazioni:
  - Quando esegui l'aggiornamento del componente aggiuntivo Amazon EKS, scegli di sovrascrivere le impostazioni esistenti come opzione di risoluzione dei conflitti. Se hai effettuato altre impostazioni personalizzate su Implementazione, assicurati di eseguire il backup delle impostazioni prima dell'aggiornamento in modo da poter riapplicare le altre impostazioni personalizzate dopo l'aggiornamento.
  - Rimuovi `PodDisruptionBudget` esistente e riprova a eseguire l'aggiornamento.
- Nelle versioni aggiuntive EKS `v1.9.3-eksbuild.3` e successive e `v1.10.1-eksbuild.6` e successive, Implementazione di CoreDNS imposta il `readinessProbe` per utilizzare l'endpoint `/ready`. Questo endpoint è abilitato nel file di configurazione `Corefile` per CoreDNS.

  Se utilizzi un dispositivo personalizzato `Corefile`, devi aggiungere il plug-in `ready` alla configurazione, in modo che l'endpoint `/ready` sia attivo in CoreDNS e possa essere utilizzato dalla sonda.
- Nelle versioni aggiuntive EKS `v1.9.3-eksbuild.7` e successive e `v1.10.1-eksbuild.4` e successive, è possibile modificare il. `PodDisruptionBudget` È possibile modificare il componente aggiuntivo e modificare queste impostazioni nelle configurazioni facoltative utilizzando i campi dell'esempio seguente. Questo esempio mostra l'impostazione predefinita `PodDisruptionBudget`.

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
}
```

Puoi impostare `maxUnavailable` o `minAvailable`, ma non puoi impostare entrambi sullo stesso `PodDisruptionBudget`. Per ulteriori informazioni`PodDisruptionBudgets`, consulta Specificare un PodDisruptionBudget nella documentazione di Kubernetes.

Tieni presente che, se imposti `enabled` su `false`, il `PodDisruptionBudget` non viene rimosso. Dopo aver impostato questo campo su `false`, è necessario eliminare l'oggetto `PodDisruptionBudget`. Allo stesso modo, se modifichi il componente aggiuntivo per utilizzarne una versione precedente (esegui il downgrade del componente aggiuntivo) dopo l'aggiornamento a una versione con un `PodDisruptionBudget`, questo non viene rimosso `PodDisruptionBudget`. Esegui il seguente comando per eliminare il `PodDisruptionBudget`:

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- Nelle versioni dei componenti aggiuntivi di EKS `v1.10.1-eksbuild.5` e successive, modifica la tolleranza predefinita da `node-role.kubernetes.io/master:NoSchedule` a `node-role.kubernetes.io/control-plane:NoSchedule` per conformarla a KEP 2067. Per ulteriori informazioni su KEP 2067, vedi KEP-2067: Rinominare l'etichetta «master» e il taint di kubeadm nelle Kubernetes Enhancement Proposals (KEP) su GitHub.

  Nelle versioni dei componenti aggiuntivi di EKS `v1.8.7-eksbuild.8` e successive e `v1.9.3-eksbuild.9` e successive, entrambe le tolleranze sono impostate per essere compatibili con tutte le versioni di Kubernetes.

- Nelle versioni aggiuntive EKS `v1.9.3-eksbuild.11` e `v1.10.1-eksbuild.7` e successive, l'implementazione di CoreDNS imposta un valore predefinito per `topologySpreadConstraints`. Il valore predefinito garantisce che i pod CoreDNS siano distribuiti tra le zone di disponibilità se sono disponibili nodi in più zone di disponibilità. Puoi impostare un valore personalizzato che verrà utilizzato al posto di quello predefinito. Il valore predefinito è riportato di seguito:

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
```

```
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

- Configurazioni Corefile personalizzate e aggiornamenti aggiuntivi: il componente aggiuntivo Amazon EKS gestisce completamente CoreDNS (). ConfigMap `data.Corefile` Se modifichi `Corefile` direttamente (ad esempio, per aggiungere server d'inoltro condizionali per la risoluzione DNS locale), il componente aggiuntivo sovrascrive le modifiche durante l'aggiornamento successivo. Per preservare le `Corefile` configurazioni personalizzate negli aggiornamenti dei componenti aggiuntivi, utilizza la chiave nel parametro del componente aggiuntivo. `corefile` `configurationValues` Per ulteriori informazioni, consulta the section called "Campi che puoi personalizzare".

Considerazioni sull'aggiornamento a **`CoreDNS v1.11`**

- Nelle versioni `v1.11.1-eksbuild.4` e successive del componente aggiuntivo di EKS, l'immagine del container è basata su un'immagine di base minima gestita da Amazon EKS Distro, che contiene pacchetti minimi e non ha shell. Per ulteriori informazioni, consulta Amazon EKS Distro. L'utilizzo e la risoluzione dei problemi dell'immagine CoreDNS rimangono gli stessi.

Creazione del componente aggiuntivo CoreDNS di Amazon EKS

Creazione del componente aggiuntivo CoreDNS di Amazon EKS. Prima di creare il componente aggiuntivo, è necessario disporre di un cluster. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster.

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.10.1-eksbuild.13
   ```

2. Scopri qual è il tipo di componente aggiuntivo attualmente installato sul cluster. A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

Se viene restituito un numero di versione, sul cluster è installato il componente aggiuntivo del tipo Amazon EKS e non è necessario completare i passaggi rimanenti di questa procedura. Se viene restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completa i passaggi rimanenti di questa procedura per installarlo.

3. Salva la configurazione del componente aggiuntivo attualmente installato.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. Creare il componente aggiuntivo utilizzando AWS CLI. Se desideri utilizzare la Console di gestione AWS o il `eksctl` per creare il componente aggiuntivo, consulta the section called "Creare un componente aggiuntivo" e indica `coredns` come nome del componente aggiuntivo. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.
   - Sostituisci *`my-cluster`* con il nome del tuo cluster.
   - Sostituire *`v1.11.3-eksbuild.1`* con la versione più recente indicata nella tabella delle versioni più recenti per la versione del cluster.

     ```
     aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version
      v1.11.3-eksbuild.1
     ```

     Se al componente aggiuntivo attuale sono state applicate impostazioni personalizzate in conflitto con le impostazioni predefinite del componente aggiuntivo di Amazon EKS, la creazione potrebbe non andare a buon fine. Se la creazione non riesce, riceverai un errore che può aiutare a risolvere il problema. In alternativa, puoi aggiungere `--resolve-conflicts OVERWRITE` al comando precedente. Ciò consente al componente aggiuntivo di sovrascrivere le impostazioni personalizzate esistenti. Una volta creato il componente aggiuntivo, è possibile aggiornarlo con impostazioni personalizzate.
5. Verificare che la versione più recente del componente aggiuntivo per la versione Kubernetes del cluster sia stata aggiunta al cluster. Sostituisci *`my-cluster`* con il nome del tuo cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

La creazione del componente aggiuntivo potrebbe richiedere alcuni secondi.

Di seguito viene riportato un output di esempio:

```
v1.11.3-eksbuild.1
```

6. Se hai creato impostazioni personalizzate per il componente aggiuntivo originale, prima di creare il componente aggiuntivo di Amazon EKS, utilizza la configurazione che hai salvato precedentemente per aggiornare il componente aggiuntivo di Amazon EKS con le tue impostazioni personalizzate. Per istruzioni sull'aggiornamento del componente aggiuntivo, consultare la pagina the section called "Aggiornamento (componente aggiuntivo EKS)".

## Aggiornamento del componente aggiuntivo CoreDNS di Amazon EKS

Aggiorna il componente aggiuntivo del tipo Amazon EKS. Se al cluster non è stato aggiunto il componente aggiuntivo Amazon EKS, aggiungerlo o consultare the section called "Aggiornamento (autogestito)".

Prima di iniziare, consultare le considerazioni sull'aggiornamento. Per ulteriori informazioni, consulta the section called "Considerazioni importanti sull'aggiornamento di CoreDNS".

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. Sostituire *my-cluster* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 "addon.addonVersion" --output text
```

Di seguito viene riportato un output di esempio:

```
v1.10.1-eksbuild.13
```

Se la versione restituita è la stessa della versione Kubernetes del cluster presente nella tabella delle versioni più recenti, la versione più recente è già installata nel cluster e non è necessario completare il resto di questa procedura. Se nell'output viene restituito un errore invece di un numero di versione, il componente aggiuntivo del tipo Amazon EKS non è installato sul cluster. È necessario creare il componente aggiuntivo prima di poterlo aggiornare con questa procedura.

2. Salva la configurazione del componente aggiuntivo attualmente installato.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. Aggiornare il componente aggiuntivo utilizzando AWS CLI. Se desideri utilizzare la Console di gestione AWS o il `eksctl` per aggiornare il componente aggiuntivo, consulta the section called "Aggiornamento di un componente aggiuntivo". Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.
   - Sostituisci *`my-cluster`* con il nome del tuo cluster.
   - Sostituire *`v1.11.3-eksbuild.1`* con la versione più recente indicata nella tabella delle versioni più recenti per la versione del cluster.
   - L'opzione `--resolve-conflicts`*`PRESERVE`* mantiene i valori di configurazione esistenti per il componente aggiuntivo. Se sono stati configurati valori personalizzati per le impostazioni del componente aggiuntivo e non si utilizza questa opzione, Amazon EKS sovrascrive i valori personalizzati con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `OVERWRITE`, tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se sono stati configurati valori personalizzati per un'impostazione qualunque, è possibile che vengano sovrascritti con i valori predefiniti di Amazon EKS. Se questo valore viene modificato in `none`, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non andare a buon fine. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.
   - Se non viene aggiornata un'impostazione di configurazione, rimuovere `--configuration-values '{`*`"replicaCount":3`*`}'` dal comando. Se viene aggiornata un'impostazione di configurazione, sostituire *`"replicaCount":3`* con l'impostazione desiderata. In questo esempio, il numero di repliche di CoreDNS è impostato su `3`. Il valore specificato deve essere valido per lo schema di configurazione. Se non si conosce lo schema di configurazione, è possibile eseguire `aws eks describe-addon-configuration --addon-name coredns --addon-version` *`v1.11.3-eksbuild.1`* sostituendo *`v1.11.3-eksbuild.1`* con il numero di versione del componente aggiuntivo di cui si desidera visualizzare la configurazione. Lo schema viene restituito nell'output. Se disponi di una configurazione personalizzata, desideri rimuoverla e reimpostare i valori di tutte le impostazioni ai valori predefiniti di Amazon EKS, rimuovi *`"replicaCount":3`* dal comando in modo da avere delle `{}` vuote. Per ulteriori informazioni sulle impostazioni di CoreDNS, consultare la pagina Customizing DNS Service della documentazione di Kubernetes.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

4. Conferma che la versione del componente aggiuntivo sia stata aggiornata. Sostituisci *my-cluster* con il nome del tuo cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

Di seguito viene riportato un output di esempio:

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.11.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

## Aggiornamento del componente aggiuntivo CoreDNS autogestito di Amazon EKS

> **Important**
>
> Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. In caso di dubbi sulle differenze tra i due tipi, consultare la pagina the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni

sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non è possibile utilizzare il componente aggiuntivo di Amazon EKS, si consiglia di segnalare il problema nel repository GitHub delle roadmap dei container.

Prima di iniziare, consultare le considerazioni sull'aggiornamento. Per ulteriori informazioni, consulta the section called "Considerazioni importanti sull'aggiornamento di CoreDNS".

1. Verifica che sul cluster sia installato il componente aggiuntivo del tipo autogestito. Sostituisci *my-cluster* con il nome del tuo cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se viene restituito un messaggio di errore, sul cluster è installato il componente aggiuntivo del tipo autogestito. Completa i passaggi rimanenti di questa procedura. Se viene restituito il numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo. Per aggiornare il componente aggiuntivo del tipo Amazon EKS, utilizzare la procedura descritta alla pagina Update the CoreDNS Amazon EKS add-on anziché quella qui indicata. In caso di dubbi sulle differenze tra i tipi di componente aggiuntivo, consultare la pagina the section called "Componenti aggiuntivi Amazon EKS".
2. Scopri qual è la versione dell'immagine di container attualmente installata sul cluster.

   ```
   kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.8.7-eksbuild.2
   ```
3. Se la versione corrente di CoreDNS è `v1.5.0` o successiva, ma precedente alla versione indicata nella tabella Versioni CoreDNS, saltare questa fase. Se la versione corrente è precedente alla `1.5.0`, è necessario modificare `ConfigMap` per CoreDNS al fine di utilizzare il componente aggiuntivo avanzato, anziché il componente aggiuntivo proxy.

   a. Aprire il file `ConfigMap` con il comando seguente:

      ```
      kubectl edit configmap coredns -n kube-system
      ```

b. Sostituire `proxy` nella linea seguente con `forward`. Salvare il file e uscire dall'editor.

```
proxy . /etc/resolv.conf
```

4. Se il cluster è stato originariamente implementato su Kubernetes `1.17` o versioni precedenti, potrebbe essere necessario rimuovere una riga obsoleta dal manifesto CoreDNS.

> **Important**
>
> È necessario completare questo passaggio prima di eseguire l'aggiornamento alla versione `1.7.0` di CoreDNS, ma è preferibile completarlo anche si esegue l'aggiornamento a una versione precedente.

a. Controllare se il manifesto CoreDNS presenta la riga di inserimento.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
 grep upstream
```

Se non viene restituito alcun output, nel manifesto non è presente la riga ed è possibile passare direttamente alla fase successiva per aggiornare CoreDNS. Se viene restituito l'output, è necessario rimuovere la riga di inserimento.

b. Modificare `ConfigMap` con il seguente comando, rimuovendo la riga nel file che ha la parola `upstream` in esso. Non apportare ulteriori modifiche al file. Una volta rimossa la riga di inserimento, salvare le modifiche.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. Recuperare la versione attuale dell'immagine CoreDNS:

```
kubectl describe deployment coredns -n kube-system | grep Image
```

Di seguito viene riportato un output di esempio:

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. Se si esegue l'aggiornamento a CoreDNS versione `1.8.3` o successive, è necessario aggiungere l'autorizzazione `endpointslices` al `system:coredns` del `clusterrole` Kubernetes.

```
kubectl edit clusterrole system:coredns -n kube-system
```

Aggiungere le seguenti righe sotto le righe di autorizzazioni esistenti nella sezione `rules` del file.

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. Aggiornare il componente aggiuntivo CoreDNS sostituendo*602401143452* e *region-code* con i valori dell'output restituiti in una fase precedente. Sostituire *v1.11.3-eksbuild.1* con la versione CoreDNS indicata nella tabella delle versioni più recenti per la versione Kubernetes.

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.3-
eksbuild.1
```

Di seguito viene riportato un output di esempio:

```
deployment.apps/coredns image updated
```

8. Controlla nuovamente la versione dell'immagine del container per accertarti che sia stata aggiornata alla versione specificata nel passaggio precedente.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

Di seguito viene riportato un output di esempio:

```
v1.11.3-eksbuild.1
```

## Scalare i pod CoreDNS per un traffico DNS elevato

Quando avvii un cluster Amazon EKS con almeno un nodo, per impostazione predefinita viene implementata una distribuzione di due repliche dell'immagine CoredNS, indipendentemente dal numero di nodi distribuiti nel cluster. I contenitori CoreDNS forniscono la risoluzione dei nomi per tutti i pod del cluster. Le applicazioni utilizzano la risoluzione dei nomi per connettersi a pod e servizi nel cluster, nonché a servizi esterni al cluster. Con l'aumentare del numero di richieste di risoluzione dei nomi (query) provenienti dai pod, i pod CoreDNS possono sovraccaricarsi, rallentare e rifiutare le richieste che i pod non sono in grado di gestire.

Per gestire il carico maggiore sui pod CoreDNS, valutare l'implementazione di un sistema di dimensionamento automatico per CoreDNS. Amazon EKS può gestire la scalabilità automatica dell'implementazione CoreDNS nella versione EKS di CoreDNS. Add-on Questo autoscaler di CoreDNS monitora continuamente lo stato del cluster, incluso il numero di nodi e core CPU. Sulla base di tali informazioni, il controller adatterà dinamicamente il numero di repliche dell'implementazione CoreDNS in un cluster EKS. Questa funzionalità è disponibile per CoreDNS versione `v1.9` e successive. Per ulteriori informazioni sulle versioni compatibili con il dimensionamento automatico di CoreDNS, consultare la sezione di seguito.

Il sistema gestisce automaticamente le repliche CoredNS utilizzando una formula dinamica basata sia sul numero di nodi che di core CPU nel cluster, calcolato come il massimo di (numero diviso per 16) e (OfNodes numero di core CPU diviso per 256). Valuta la domanda su periodi di picco di 10 minuti e aumenta verticalmente in modo immediato quando necessario per gestire l'aumento del carico di query DNS, mentre riduce verticalmente in modo graduale tramite la riduzione delle repliche del 33% ogni 3 minuti per mantenere la stabilità del sistema ed evitare interruzioni.

Consigliamo di utilizzare questa funzionalità insieme ad altre best practice di dimensionamento automatico dei cluster EKSper migliorare la disponibilità dell'applicazione e la scalabilità del cluster in generale.

### Prerequisiti

Esistono tre prerequisiti affinché Amazon EKS possa scalare l'implementazione CoreDNS:

- È necessario utilizzare la Add-on versione EKS di CoredNS.
- Il cluster deve eseguire almeno le versioni minime del cluster e della piattaforma.
- Il tuo cluster deve eseguire almeno la versione minima di EKS Add-on di CoredNS.

Versione minima del cluster

Il dimensionamento automatica di CoreDNS viene eseguito da un nuovo componente nel piano di controllo (control-plane) del cluster, gestito da Amazon EKS. Per questo, è necessario aggiornare il cluster a una versione EKS che supporti la versione minima della piattaforma che include il nuovo componente.

Un nuovo cluster Amazon EKS. Per implementarne uno, consulta Nozioni di base. Il cluster deve eseguire una delle versioni di Kubernetes e della piattaforma elencate nella tabella seguente o versioni successive. Sono supportate anche tutte le versioni di Kubernetes e della piattaforma successive a quelle elencate. Puoi controllare la tua versione attuale di Kubernetes *my-cluster* sostituendo il seguente comando con il nome del cluster e quindi eseguendo il comando modificato:

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

| Versione di Kubernetes | Versione della piattaforma |
|---|---|
| Non elencata | Tutte le versioni della piattaforma |
| 1.29.3 | eks.7 |
| 1.28.8 | eks.13 |
| 1.27.12 | eks.17 |
| 1.26.15 | eks.18 |

> **Note**
>
> Sono supportate anche tutte le versioni della piattaforma delle versioni successive di Kubernetes, ad esempio la versione Kubernetes 1.30 da eks.1 in poi.

Versione minima EKS Add-on

| Versione di Kubernetes | 1.29 | 1,28 |
|---|---|---|
| | v1.11.1-eksbuild.9 | v1.10.1-eksbuild.11 |

Configurazione della scalabilità automatica di CoredNS in Console di gestione AWS

1. Assicurarsi che il cluster disponga della versione minima del cluster o successiva.

   Amazon EKS aggiorna automaticamente i cluster tra le versioni della piattaforma della stessa versione di Kubernetes e non è possibile avviare questo processo autonomamente. È invece possibile aggiornare il cluster alla versione successiva di Kubernetes e ciò comporterà l'aggiornamento del cluster alla versione K8s e alla versione della piattaforma più recente.

   Le nuove versioni di Kubernetes hanno introdotto modifiche significative. Pertanto consigliamo di verificare il comportamento delle applicazioni utilizzando un cluster separato con la nuova versione di Kubernetes prima di eseguire l'aggiornamento sui cluster di produzione.

   Per aggiornare un cluster a una nuova versione di Kubernetes, seguire la procedura descritta alla pagina Update existing cluster to new Kubernetes version.
2. Assicurati di avere EKS Add-on for CoreDNS, non il sistema CoredNS Deployment autogestito.

   A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Eseguire il comando seguente per verificare quale tipo di componente aggiuntivo è installato sul cluster. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se viene restituito un numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo ed è possibile procedere al passaggio successivo. Se viene restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completare i passaggi rimanenti della procedura descritta alla pagina Create the CoreDNS Amazon EKS add-on per sostituire la versione autogestita con il componente aggiuntivo EKS.
3. Assicurati che EKS Add-on for CoredNS abbia una versione uguale o superiore alla versione EKS minima. Add-on

   Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. Puoi effettuare il check-in Console di gestione AWS o eseguire il seguente comando:

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

Di seguito viene riportato un output di esempio.

```
v1.10.1-eksbuild.13
```

Confronta questa versione con la Add-on versione EKS minima nella sezione precedente. Se necessario, aggiorna EKS Add-on a una versione superiore seguendo la procedura Aggiorna il componente aggiuntivo CoredNS Amazon EKS.

4. Aggiungi la configurazione di scalabilità automatica alle impostazioni di configurazione opzionali di EKS. Add-on
   a. Aprire la Console Amazon EKS.
   b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui configurare il componente aggiuntivo.
   c. Scegli la scheda Add-ons.
   d. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo CoreDNS e scegliere Modifica.
   e. Nella pagina Configurazione di CoreDNS:
      i. Selezionare la Versione da utilizzare. Si consiglia di mantenere la stessa versione rispetto al passaggio precedente e di aggiornare la versione e la configurazione con operazioni separate.
      ii. Scegliere Impostazioni di configurazione facoltative.
      iii. Inserire la chiave JSON `"autoscaling":` e il valore di un oggetto JSON annidato con una chiave `"enabled":` e un valore `true` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`. Il seguente esempio mostra che il dimensionamento automatico è abilitato:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

      iv. (Facoltativo) È possibile fornire valori minimi e massimi a cui il dimensionamento automatico può scalare il numero di pod CoreDNS.

Il seguente esempio mostra che il dimensionamento automatico è abilitato e tutte le chiavi facoltative hanno dei valori. Si consiglia di utilizzare un numero minimo di pod CoreDNS sempre superiore a 2 per garantire la resilienza del servizio DNS nel cluster.

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. Scegliere Salva modifiche per applicare la nuova configurazione sostituendo i pod CoreDNS.

Amazon EKS applica le modifiche a EKS Add-ons utilizzando un'implementazione di Kubernetes Deployment for CoredNS. Puoi tenere traccia dello stato dell'implementazione nella cronologia degli aggiornamenti del componente aggiuntivo in e con. Console di gestione AWS `kubectl rollout status deployment/coredns --namespace kube-system`

`kubectl rollout` dispone dei seguenti comandi:

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, iniziare dalla cronologia del rollout ed eseguire `kubectl logs` su un pod CoreDNS per visualizzare i log di CoreDNS.

5. Se la nuova voce nella Cronologia degli aggiornamenti ha lo stato Operazione riuscita, il rollout è stato completato e il componente aggiuntivo utilizza la nuova configurazione in tutti i pod CoreDNS. Con la modifica del numero di nodi e core CPU dei nodi nel cluster, Amazon EKS scala il numero di repliche dell'implementazione CoreDNS.

Configurazione della scalabilità automatica di CoredNS in AWS Interfaccia a riga di comando

1. Assicurarsi che il cluster disponga della versione minima del cluster o successiva.

   Amazon EKS aggiorna automaticamente i cluster tra le versioni della piattaforma della stessa versione di Kubernetes e non è possibile avviare questo processo autonomamente. È invece possibile aggiornare il cluster alla versione successiva di Kubernetes e ciò comporterà l'aggiornamento del cluster alla versione K8s e alla versione della piattaforma più recente.

   Le nuove versioni di Kubernetes hanno introdotto modifiche significative. Pertanto consigliamo di verificare il comportamento delle applicazioni utilizzando un cluster separato con la nuova versione di Kubernetes prima di eseguire l'aggiornamento sui cluster di produzione.

   Per aggiornare un cluster a una nuova versione di Kubernetes, seguire la procedura descritta alla pagina Update existing cluster to new Kubernetes version.
2. Assicurati di avere EKS Add-on for CoreDNS, non il sistema CoredNS Deployment autogestito.

   A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Eseguire il comando seguente per verificare quale tipo di componente aggiuntivo è installato sul cluster. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se viene restituito il numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo. Se viene restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completare i passaggi rimanenti della procedura descritta alla pagina Create the CoreDNS Amazon EKS add-on per sostituire la versione autogestita con il componente aggiuntivo EKS.
3. Assicurati che EKS Add-on for CoredNS abbia una versione uguale o superiore alla versione EKS minima. Add-on

   Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. Puoi effettuare il check-in Console di gestione AWS o eseguire il seguente comando:

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

Di seguito viene riportato un output di esempio.

```
v1.10.1-eksbuild.13
```

Confronta questa versione con la Add-on versione EKS minima nella sezione precedente. Se necessario, aggiorna EKS Add-on a una versione superiore seguendo la procedura Aggiorna il componente aggiuntivo CoredNS Amazon EKS.

4. Aggiungi la configurazione di scalabilità automatica alle impostazioni di configurazione opzionali di EKS. Add-on

   Esegui il seguente comando AWS CLI. Sostituire `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che si sta utilizzando.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

   Amazon EKS applica le modifiche a EKS Add-ons utilizzando un'implementazione di Kubernetes Deployment for CoredNS. Puoi tenere traccia dello stato dell'implementazione nella cronologia degli aggiornamenti del componente aggiuntivo in e con. Console di gestione AWS `kubectl rollout status deployment/coredns --namespace kube-system`

   `kubectl rollout` dispone dei seguenti comandi:

   ```
   kubectl rollout

   history  -- View rollout history
   pause    -- Mark the provided resource as paused
   restart  -- Restart a resource
   resume   -- Resume a paused resource
   status   -- Show the status of the rollout
   undo     -- Undo a previous rollout
   ```

   Se il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, iniziare dalla cronologia del rollout ed eseguire `kubectl logs` su un pod CoreDNS per visualizzare i log di CoreDNS.

5. (Facoltativo) È possibile fornire valori minimi e massimi a cui il dimensionamento automatico può scalare il numero di pod CoreDNS.

   Il seguente esempio mostra che il dimensionamento automatico è abilitato e tutte le chiavi facoltative hanno dei valori. Si consiglia di utilizzare un numero minimo di pod CoreDNS sempre superiore a 2 per garantire la resilienza del servizio DNS nel cluster.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true,"minReplicas":2,"maxReplicas":10}}'
   ```

6. Verificare lo stato dell'aggiornamento del componente aggiuntivo eseguendo il seguente comando:

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns
   ```

   Se viene visualizzata la riga `"status": "ACTIVE"`, il rollout è stato completato e il componente aggiuntivo utilizza la nuova configurazione in tutti i pod CoreDNS. Con la modifica del numero di nodi e core CPU dei nodi nel cluster, Amazon EKS scala il numero di repliche dell'implementazione CoreDNS.

Monitorare la risoluzione DNS di Kubernetes con parametri CoreDNS

CoreDNS come componente aggiuntivo EKS espone i parametri di CoreDNS sulla porta `9153` nel formato Prometheus nel servizio `kube-dns`. Puoi usare Prometheus, l'agente Amazon CloudWatch o qualsiasi altro sistema compatibile per eseguire lo scraping (raccogliere) questi parametri.

Per un esempio di configurazione di scraping compatibile sia con Prometheus che con l'agente CloudWatch, consulta Configurazione dell'agente CloudWatch per Prometheus nella Guida per l'utente di Amazon CloudWatch.

## Gestisci **`kube-proxy`** nei cluster Amazon EKS

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non è necessario installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. Se non sono ben chiare le differenze tra i due tipi, consulta the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non riesci a utilizzare il componente aggiuntivo Amazon EKS, ti invitiamo a segnalare il motivo per cui non puoi farlo al repository delle roadmap di Containers. GitHub

Il componente aggiuntivo `kube-proxy` viene implementato su ogni nodo Amazon EC2 del cluster Amazon EKS. Mantiene le regole di rete sui nodi e consente la comunicazione di rete con i tuoi pod. Il componente aggiuntivo non viene implementato sui nodi Fargate nel cluster. Per ulteriori informazioni, consulta kube-proxy nella documentazione Kubernetes.

## Installa come Amazon EKS Add-on

### `versioni kube-proxy`

La tabella seguente elenca la versione più recente del componente aggiuntivo di Amazon EKS disponibile per ogni versione di Kubernetes.

| Versione di Kubernetes | Versione **`kube-proxy`** |
|---|---|
| 1.36 | v1.36.0-eksbuild.17 |
| 1,35 | v1.35.3-eksbuild.21 |
| 1,34 | v1.34.6-eksbuild.21 |
| 1.33 | v1.33.10-eksbuild.21 |
| 1,32 | v1.32.13-eksbuild.24 |
| 1,31 | v1.31.14-eksbuild.28 |

> **Note**
>
> Se gestisci autonomamente questo componente aggiuntivo, le versioni nella tabella potrebbero non essere le stesse delle versioni autogestite disponibili.

## immagine del contenitore kube-proxy

L'immagine del container `kube-proxy` è basata su un'immagine di base minima gestita da Amazon EKS Distro, che contiene pacchetti minimi e non ha shell. Per ulteriori informazioni, consulta Amazon EKS Distro.

La tabella seguente elenca la versione più recente disponibile dell'immagine del container `kube-proxy` auto-gestita per ciascuna versione del cluster Amazon EKS.

| Versione | kube-proxy |
|---|---|
| 1.36 | v1.36.0-eksbuild.17 |
| 1,35 | v1.35.3-eksbuild.21 |
| 1,34 | v1.34.6-eksbuild.21 |
| 1.33 | v1.33.10-minimal-eksbuild.21 |
| 1,32 | v1.32.13-minimal-eksbuild.24 |
| 1,31 | v1.31.14-minimal-eksbuild.28 |

Quando aggiorni un tipo di componente aggiuntivo Amazon EKS, specifichi una versione del componente aggiuntivo Amazon EKS valida, che potrebbe non essere una versione elencata in questa tabella. Questo accade perché le versioni dei componenti aggiuntivi di Amazon EKS non sempre corrispondono a quelle delle immagini del container specificate durante l'aggiornamento del componente aggiuntivo del tipo autogestito. Quando si aggiorna il componente aggiuntivo del tipo autogestito, si specifica una versione valida dell'immagine di container indicata in questa tabella.

### Aggiornare il componente aggiuntivo autogestito **kube-proxy** Kubernetes

> **Important**
>
> Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. Se non conosci bene le differenze tra i due tipi, consulta la pagina the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non riesci a utilizzare il componente aggiuntivo

di Amazon EKS, ti invitiamo a segnalare il problema in Containers roadmap GitHub repository.

Prerequisiti

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.

Considerazioni

- Kube-proxy su un cluster Amazon EKS ha la stessa policy di compatibilità e disallineamento di Kubernetes. Scopri Verifying Amazon EKS add-on version compatibility with a cluster.
  1. Verifica che sul cluster sia installato il componente aggiuntivo del tipo autogestito. Sostituisci *my-cluster* con il nome del tuo cluster.

     ```
     aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
      addon.addonVersion --output text
     ```

     Se viene restituito un messaggio di errore, sul cluster è installato il componente aggiuntivo del tipo autogestito. I passaggi rimanenti di questo argomento riguardano l'aggiornamento del componente aggiuntivo del tipo autogestito. Se viene restituito il numero di versione, sul cluster è installato il componente aggiuntivo del tipo Amazon EKS. Per aggiornarlo, utilizza la procedura descritta in Updating an Amazon EKS add-on anziché quella presente in questo argomento. Se non conosci bene le differenze tra i due tipi di componente aggiuntivo, consulta the section called "Componenti aggiuntivi Amazon EKS".
  2. Scopri qual è la versione dell'immagine di container attualmente installata sul cluster.

     ```
     kubectl describe daemonset kube-proxy -n kube-system | grep Image
     ```

     Di seguito viene riportato un output di esempio:

     ```
     Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-
     eksbuild.2
     ```

     Nell'output di esempio, *v1.29.1-eksbuild.2* è la versione installata sul cluster.
  3. Aggiorna il componente aggiuntivo kube-proxy sostituendo *602401143452* e *region-code* con i valori dell'output del passaggio precedente. Sostituisci *v1.30.6-eksbuild.3* con la

versione `kube-proxy` riportata nella tabella Latest available self-managed kube-proxy container image version for each Amazon EKS cluster version.

> **Important**
>
> I manifesti per ogni tipo di immagine sono diversi e non compatibili tra i tipi di immagine predefiniti o minimi. È necessario utilizzare lo stesso tipo di immagine di quella precedente, in modo che il punto di ingresso e gli argomenti corrispondano.

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.6-
eksbuild.3
```

Di seguito viene riportato un output di esempio:

```
daemonset.apps/kube-proxy image updated
```

4. Verifica che la nuova versione sia ora installata nel cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

Di seguito viene riportato un output di esempio:

```
v1.30.0-eksbuild.3
```

5. Se utilizzi nodi `x86` e `Arm` nello stesso cluster e il cluster è stato implementato il prima del 17 agosto 2020. Modificare, il manifesto `kube-proxy` attraverso il seguente comando, per includere un selettore di nodi per più architetture hardware. Questa è un'operazione da compiere una sola volta. Dopo aver aggiunto il selettore al manifesto, non occorrerà aggiungerlo ogni volta che aggiorni il componente aggiuntivo. Se il cluster è stato implementato in data 17 agosto 2020 o successiva, il `kube-proxy` è già in grado di supportare una multi-architettura.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Aggiungere il seguente selettore di nodo al file nell'editor, quindi salvare il file. Per un esempio di dove includere questo testo nell'editor, vedere manifesto CNI su GitHub. Ciò consente a Kubernetes di estrarre l'immagine hardware corretta in base all'architettura hardware del nodo.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. Se il cluster è stato originariamente creato con Kubernetes `1.14` o versione successiva, è possibile saltare questo passaggio perché `kube-proxy` include già `Affinity Rule`. Se in origine hai creato un cluster Amazon EKS con la versione `1.13` di Kubernetes o precedente e intendi utilizzare nodi Fargate nel tuo cluster, modifica il manifesto `kube-proxy` in modo da includere una regola `NodeAffinity` per impedire ai pod `kube-proxy` la pianificazione su nodi Fargate. Questa modifica si applica una sola volta. Dopo aver aggiunto `Affinity Rule` al manifesto, non occorrerà aggiungerlo ogni volta che aggiorni il componente aggiuntivo. Modificare il DaemonSet `kube-proxy`.

   ```
   kubectl edit -n kube-system daemonset/kube-proxy
   ```

   Aggiungi i seguenti elementi `Affinity Rule` alla sezione DaemonSet `spec` del file nell'editor e salvare il file. Per un esempio di dove includere questo testo nell'editor, vedere manifesto CNI su GitHub.

   ```
   - key: eks.amazonaws.com/compute-type
     operator: NotIn
     values:
     - fargate
   ```

# Informazioni su come implementare carichi di lavoro e componenti aggiuntivi su Amazon EKS

I carichi di lavoro vengono implementati in container, a loro volta implementati in pod in Kubernetes. Un pod include uno o più container. In genere, uno o più pod che forniscono lo stesso servizio vengono implementati in un servizio Kubernetes. Dopo aver implementato più pod che forniscono lo stesso servizio, puoi:

- Visualizzare le informazioni sui carichi di lavoro in esecuzione su ogni cluster utilizzando il Console di gestione AWS.
- Dimensionare verticalmente i pod verso l'alto o verso il basso con Vertical Pod Autoscaler di Kubernetes.
- Dimensionare orizzontalmente verso l'alto o verso il basso il numero di pod necessari per soddisfare la domanda con Horizontal Pod Autoscaler di Kubernetes.
- Creare un Network Load Balancer esterno (per i pod accessibili a internet) o interno (per i pod privati) per bilanciare il traffico di rete tra i pod. Il load balancer indirizza il traffico al livello 4 del modello OSI.
- Creare un Application Load Balancer per bilanciare il traffico delle applicazioni tra i pod. Il load balancer di applicazione indirizza il traffico al livello 7 del modello OSI.
- Se sei un nuovo utente di Kubernetes, questo argomento aiuta a implementare un'applicazione di esempio.
- È possibile limitare gli indirizzi IP assegnabili a un servizio con `externalIPs`.

## Implementazione di un'applicazione esemplificativa

In questo argomento, si distribuisce un'applicazione di esempio nel cluster su nodi Linux.

### Prerequisiti

- Un cluster Kubernetes esistente con almeno un nodo. Se non si dispone di un cluster Amazon EKS esistente, è possibile implementarne uno utilizzando una delle guide in Nozioni di base.
- `Kubectl` installato sul computer. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".

- `Kubectl` configurato per comunicare con il cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".
- Se si prevede di implementare il carico di lavoro di esempio in Fargate, è necessario disporre di un profilo Fargate che include lo stesso namespace creato in questo tutorial, che è `eks-sample-app`, a meno che non cambi il nome. Se hai creato un cluster utilizzando una delle guideNozioni di base, dovrai creare un nuovo profilo o aggiungere lo spazio dei nomi al profilo esistente, poiché il profilo creato nelle guide introduttive non specifica lo spazio dei nomi utilizzato in questo tutorial. Il VPC deve disporre di almeno una sottorete privata.

Sebbene molte variabili siano modificabili nei passaggi seguenti, si consiglia di modificare solo i valori delle variabili, se specificato. Con una migliore conoscenza dei pod, delle implementazioni e dei servizi Kubernetes, è possibile provare a modificare altri valori.

## Creare un namespace

Un namespace consente di raggruppare le risorse in Kubernetes. Per ulteriori informazioni, consulta Namespace nella documentazione Kubernetes. Se prevedi di distribuire la tua applicazione di esempio per semplificare la gestione del calcolo con AWS Fargate, assicurati che il valore di Define that Pods use Fargate `namespace` all'avvio sia. AWS `eks-sample-app`

```
kubectl create namespace eks-sample-app
```

## Creazione di un'implementazione Kubernetes

Creare un'implementazione Kubernetes. Questa implementazione esemplificativa estrae un'immagine container da un repository pubblico e ne implementa tre repliche (pod individuali) sul cluster. Per ulteriori informazioni, consultare la sezione Implementazioni nella documentazione Kubernetes.

1. Salva i contenuti seguenti in un file denominato `eks-sample-deployment.yaml`. I container nell'applicazione esemplificativa non utilizzano l'archiviazione di rete, ma potrebbe essere necessaria per determinate applicazioni. Per ulteriori informazioni, consulta Archiviazione di dati per app.
   - `amd64` o `arm64` `values` nella chiave `kubernetes.io/arch` significa che l'applicazione può essere implementata su entrambe le architetture hardware (se entrambe sono presenti nel cluster). Ciò è possibile perché questa immagine è un'immagine multi-architettura, ma non tutte lo sono. È possibile determinare l'architettura hardware su cui è supportata

l'immagine visualizzando i dettagli dell'immagine nel repository da cui la stai estraendo. Quando si implementano immagini che non supportano un tipo di architettura hardware o che non si desidera implementare, rimuovere quel tipo ti architettura dal manifesto. Per ulteriori informazioni, consulta Well-Known Labels, Annotations and Taints nella documentazione di Kubernetes.

- `kubernetes.io/os: linux` `nodeSelector` significa che se avessi nodi Linux e Windows, ad esempio nel cluster, l'immagine verrebbe implementata solo sui nodi Linux. Per ulteriori informazioni, consulta Well-Known Labels, Annotations and Taints nella documentazione di Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
```

```
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

2. Applicare il manifesto di implementazione al cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

## Creazione di un servizio

Un servizio consente di accedere a tutte le repliche tramite un unico indirizzo IP o nome. Per ulteriori informazioni, consultare la sezione Servizio nella documentazione Kubernetes. Sebbene non sia implementato nell'applicazione di esempio, se hai applicazioni che devono interagire con altri AWS servizi, ti consigliamo di creare account di servizio Kubernetes per i tuoi Pod e associarli agli account IAM. AWS Specificando gli account di servizio, è possibile che i pod dispongano delle autorizzazioni minime necessarie per interagire con altri servizi. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

1. Salva i contenuti seguenti in un file denominato `eks-sample-service.yaml`. Kubernetes assegna al servizio il proprio indirizzo IP accessibile solo dall'interno del cluster. Per accedere al servizio dall'esterno del cluster, implementare il controller del load balancer AWS per bilanciare il carico dell'applicazione o del traffico di rete verso il servizio.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Applicare il manifesto del servizio al cluster.

```
kubectl apply -f eks-sample-service.yaml
```

## Esame delle risorse create

1. Visualizzare tutte le risorse nel namespace `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Di seguito viene riportato un output di esempio.

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22     1/1     Running   0          27m

NAME                                 TYPE          CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-linux-service     ClusterIP     10.100.74.8     <none>        80/TCP
   32m

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment     3/3     3            3           27m

NAME                                                        DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8      3         3         3
 27m
```

Nell'output, è possibile visualizzare il servizio e l'implementazione specificati nel manifesto di esempio implementato nei passaggi precedenti. Verranno mostrati anche tre pod. Questo perché 3 `replicas` sono state specificate nel manifesto di esempio. Per ulteriori informazioni sui pod, consulta Pod nella documentazione Kubernetes. Kubernetes crea automaticamente la risorsa `replicaset`, anche se non è specificato nei manifesti esemplificativi. Per ulteriori informazioni in meritoReplicaSets, consulta la documentazione di ReplicaSet Kubernetes.

> **Note**
>
> Kubernetes mantiene il numero di repliche specificate nel manifesto. Se si tratta di un'implementazione di produzione e si desidera che Kubernetes scali orizzontalmente il numero di repliche e scali verticalmente le risorse di calcolo per i pod, utilizza Scale pod deployments with Horizontal Pod Autoscaler e Adjust pod resources with Vertical Pod Autoscaler per fare ciò.

2. Visualizzare i dettagli del servizio implementato.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

Di seguito viene riportato un output di esempio.

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Nell'output precedente, il valore per `IP:` è un indirizzo IP univoco che può essere raggiunto da qualsiasi nodo o pod all'interno del cluster, ma non può essere raggiunto dall'esterno del cluster. I valori per `Endpoints` sono indirizzi IP assegnati all'interno del VPC ai pod che fanno parte del servizio.

3. Visualizza i dettagli di uno dei pod elencati nell'output al momento della visualizzazione del namespace in un passaggio precedente. Sostituiscilo *`65b7669776-m6qxz`* con il valore restituito per uno dei tuoi Pod.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

Output esemplificativo abbreviato

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz to
 ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Nell'output precedente, il valore per `IP:` è un IP univoco assegnato al pod dall'intervallo CIDR assegnato alla sottorete in cui si trova il nodo. Se si preferisce che ai pod vengano assegnati indirizzi IP da blocchi CIDR diversi, è possibile modificare il comportamento di default. Per ulteriori informazioni, consulta the section called "Rete personalizzata". É anche possibile vedere come il pianificatore Kubernetes abbia pianificato il pod sul `Node` con l'indirizzo IP *`192.168.45.132`*.

> **Tip**
>
> Anziché utilizzare la riga di comando, è possibile visualizzare molti dettagli su pod, servizi, implementazioni e altre risorse Kubernetes nella Console di gestione AWS. Per ulteriori informazioni, consulta the section called "Accesso alle risorse di cluster".

# Eseguire una shell su un pod

1. Esegui una shell sul Pod che hai descritto nel passaggio precedente, sostituendola *65b7669776-m6qxz* con l'ID di uno dei tuoi Pod.

   ```
   kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app -- /
   bin/bash
   ```

2. Dalla shell del pod, visualizzare l'output dal server Web installato con l'implementazione in una fase precedente. È necessario specificare solo il nome del servizio. Ciò è risolto sull'indirizzo IP del servizio da CoreDNS, implementato con un cluster Amazon EKS, per impostazione predefinita.

   ```
   curl eks-sample-linux-service
   ```

   Di seguito viene riportato un output di esempio.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

3. Dalla shell del pod, visualizza il server DNS per il pod.

   ```
   cat /etc/resolv.conf
   ```

   Di seguito viene riportato un output di esempio.

   ```
   nameserver 10.100.0.10
   search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
   west-2.compute.internal
   options ndots:5
   ```

   Nell'output precedente, `10.100.0.10` viene assegnato automaticamente come `nameserver` per tutti i pod implementati nel cluster.

4. Disconnettersi dal pod digitando `exit`.

5. Una volta terminato l'uso dell'applicazione esemplificativa, sarà possibile rimuovere il namespace, il servizio e l'implementazione esemplificativa con il seguente comando.

```
kubectl delete namespace eks-sample-app
```

### Fasi successive

Dopo aver implementato l'applicazione esemplificativa, potresti provare a svolgere alcune delle seguenti esercitazioni:

- Esegui il routing del traffico di applicazioni e HTTP con Application Load Balancer
- Esegui il routing del traffico TCP e UDP con Network Load Balancer

## Implementazione di un'applicazione esemplificativa su Windows

In questa sezione, imparerai a implementare un'applicazione esemplificativa nel cluster sui nodi Windows.

### Prerequisiti

- Un cluster Kubernetes esistente con almeno un nodo. Se non si dispone di un cluster Amazon EKS esistente, è possibile implementarne uno utilizzando una delle guide in Nozioni di base. Devi avere il Supporto Windows abilitato per il cluster e almeno un nodo Windows di Amazon EC2.
- `Kubectl` installato sul computer. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".
- `Kubectl` configurato per comunicare con il cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".
- Se si prevede di implementare il carico di lavoro di esempio in Fargate, è necessario disporre di un profilo Fargate che include lo stesso namespace creato in questo tutorial, che è `eks-sample-app`, a meno che non cambi il nome. Se hai creato un cluster con una delle guide inNozioni di base, dovrai creare un nuovo profilo o aggiungere lo spazio dei nomi al tuo profilo esistente, perché il profilo creato nelle guide introduttive non specifica lo spazio dei nomi utilizzato in questo tutorial. Il VPC deve disporre di almeno una sottorete privata.

Sebbene molte variabili siano modificabili nei passaggi seguenti, si consiglia di modificare solo i valori delle variabili, se specificato. Con una migliore conoscenza dei pod, delle implementazioni e dei servizi Kubernetes, è possibile provare a modificare altri valori.

## Creare un namespace

Un namespace consente di raggruppare le risorse in Kubernetes. Per ulteriori informazioni, consulta Namespace nella documentazione Kubernetes. Se prevedi di distribuire la tua applicazione di esempio per semplificare la gestione del calcolo con AWS Fargate, assicurati che il valore di Define that Pods use Fargate `namespace` all'avvio sia. AWS `eks-sample-app`

```
kubectl create namespace eks-sample-app
```

## Creazione di un'implementazione Kubernetes

Questa implementazione esemplificativa estrae un'immagine container da un repository pubblico e ne implementa tre repliche (pod individuali) sul cluster. Per ulteriori informazioni, consultare la sezione Implementazioni nella documentazione Kubernetes.

1. Salva i contenuti seguenti in un file denominato `eks-sample-deployment.yaml`. I container nell'applicazione esemplificativa non utilizzano l'archiviazione di rete, ma potrebbe essere necessaria per determinate applicazioni. Per ulteriori informazioni, consulta Archiviazione di dati per app.
   - `kubernetes.io/os: windows` `nodeSelector` significa che se avessi nodi Windows e Linux, ad esempio nel cluster, l'immagine verrebbe implementata solo nei nodi Windows. Per ulteriori informazioni, consulta Well-Known Labels, Annotations and Taints nella documentazione di Kubernetes.

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: eks-sample-windows-deployment
     namespace: eks-sample-app
     labels:
       app: eks-sample-windows-app
   spec:
     replicas: 3
     selector:
       matchLabels:
         app: eks-sample-windows-app
     template:
       metadata:
         labels:
           app: eks-sample-windows-app
   ```

```
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo '<html><body><br/
><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>' > C:\\inetpub\\wwwroot\
\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

2. Applicare il manifesto di implementazione al cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

## Creazione di un servizio

Un servizio consente di accedere a tutte le repliche tramite un unico indirizzo IP o nome. Per ulteriori informazioni, consultare la sezione Servizio nella documentazione Kubernetes. Sebbene non sia implementato nell'applicazione di esempio, se hai applicazioni che devono interagire con altri AWS servizi, ti consigliamo di creare account di servizio Kubernetes per i tuoi Pod e associarli agli account IAM. AWS Specificando gli account di servizio, è possibile che i pod dispongano delle autorizzazioni minime necessarie per interagire con altri servizi. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

1. Salva i contenuti seguenti in un file denominato `eks-sample-service.yaml`. Kubernetes assegna al servizio il proprio indirizzo IP accessibile solo dall'interno del cluster. Per accedere al servizio dall'esterno del cluster, implementare il controller del load balancer AWS per bilanciare il carico dell'applicazione o del traffico di rete verso il servizio.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Applicare il manifesto del servizio al cluster.

```
kubectl apply -f eks-sample-service.yaml
```

## Esame delle risorse create

1. Visualizzare tutte le risorse nel namespace `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Di seguito viene riportato un output di esempio.

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-windows-deployment-65b7669776-m6qxz   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                 TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
```

```
service/eks-sample-windows-service   ClusterIP    10.100.74.8     <none>        80/
TCP    32m

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-windows-deployment 3/3     3            3           27m

NAME                                                     DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-windows-deployment-776d8f8fd8    3         3         3
   27m
```

Nell'output, è possibile visualizzare il servizio e l'implementazione specificati nel manifesto di esempio implementato nei passaggi precedenti. Verranno mostrati anche tre pod. Questo perché 3 `replicas` sono state specificate nel manifesto di esempio. Per ulteriori informazioni sui pod, consulta Pod nella documentazione Kubernetes. Kubernetes crea automaticamente la risorsa `replicaset`, anche se non è specificato nei manifesti esemplificativi. Per ulteriori informazioni in meritoReplicaSets, consulta la documentazione di ReplicaSet Kubernetes.

> **Note**
>
> Kubernetes mantiene il numero di repliche specificate nel manifesto. Se si tratta di un'implementazione di produzione e si desidera che Kubernetes scali orizzontalmente il numero di repliche e scali verticalmente le risorse di calcolo per i pod, utilizza Scale pod deployments with Horizontal Pod Autoscaler e Adjust pod resources with Vertical Pod Autoscaler per fare ciò.

2. Visualizzare i dettagli del servizio implementato.

```
kubectl -n eks-sample-app describe service eks-sample-windows-service
```

Di seguito viene riportato un output di esempio.

```
Name:              eks-sample-windows-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-windows-app
Annotations:       <none>
Selector:          app=eks-sample-windows-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
```

```
IPs:                10.100.74.8
Port:               <unset>  80/TCP
TargetPort:         80/TCP
Endpoints:          192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:   None
Events:             <none>
```

Nell'output precedente, il valore per `IP:` è un indirizzo IP univoco che può essere raggiunto da qualsiasi nodo o pod all'interno del cluster, ma non può essere raggiunto dall'esterno del cluster. I valori per `Endpoints` sono indirizzi IP assegnati all'interno del VPC ai pod che fanno parte del servizio.

3. Visualizza i dettagli di uno dei pod elencati nell'output al momento della visualizzazione del namespace in un passaggio precedente. Sostituiscilo *65b7669776-m6qxz* con il valore restituito per uno dei tuoi Pod.

```
kubectl -n eks-sample-app describe pod eks-sample-windows-deployment-65b7669776-m6qxz
```

Output esemplificativo abbreviato

```
Name:         eks-sample-windows-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-windows-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
```

```
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-windows-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Nell'output precedente, il valore per `IP:` è un IP univoco assegnato al pod dall'intervallo CIDR assegnato alla sottorete in cui si trova il nodo. Se si preferisce che ai pod vengano assegnati indirizzi IP da blocchi CIDR diversi, è possibile modificare il comportamento di default. Per ulteriori informazioni, consulta the section called "Rete personalizzata". É anche possibile vedere come il pianificatore Kubernetes abbia pianificato il pod sul `Node` con l'indirizzo IP *`192.168.45.132`*.

> **Tip**
>
> Anziché utilizzare la riga di comando, è possibile visualizzare molti dettagli su pod, servizi, implementazioni e altre risorse Kubernetes nella Console di gestione AWS. Per ulteriori informazioni, consulta the section called "Accesso alle risorse di cluster".

## Eseguire una shell su un pod

1. Esegui una shell sul Pod che hai descritto nel passaggio precedente, sostituendola *`65b7669776-m6qxz`* con l'ID di uno dei tuoi Pod.

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app --
 powershell.exe
```

2. Dalla shell del pod, visualizzare l'output dal server Web installato con l'implementazione in una fase precedente. È necessario specificare solo il nome del servizio. Ciò è risolto sull'indirizzo IP del servizio da CoreDNS, implementato con un cluster Amazon EKS, per impostazione predefinita.

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

Di seguito viene riportato un output di esempio.

```
StatusCode        : 200
StatusDescription : OK
Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e > < H
 1 > H e l l o
```

```
                    E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t m l >
```

3. Dalla shell del pod, visualizza il server DNS per il pod.

```
Get-NetIPConfiguration
```

   Output abbreviato

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

   Nell'output precedente, `10.100.0.10` viene assegnato automaticamente come il server DNS per tutti i pod implementati nel cluster.

4. Disconnettersi dal pod digitando `exit`.
5. Una volta terminato l'uso dell'applicazione esemplificativa, sarà possibile rimuovere il namespace, il servizio e l'implementazione esemplificativa con il seguente comando.

```
kubectl delete namespace eks-sample-app
```

## Fasi successive

Dopo aver implementato l'applicazione esemplificativa, potresti provare a svolgere alcune delle seguenti esercitazioni:

- Esegui il routing del traffico di applicazioni e HTTP con Application Load Balancer
- Esegui il routing del traffico TCP e UDP con Network Load Balancer

# Regolazione delle risorse del pod con Vertical Pod Autoscaler

Vertical Pod Autoscaler di Kubernetes regola automaticamente le prenotazioni di CPU e memoria per i pod per aiutare a "dimensionare correttamente" le applicazioni. Questo processo di adeguamento può migliorare l'utilizzo delle risorse del cluster e liberare CPU e memoria per altri pod. Questo argomento illustra come implementare Vertical Pod Autoscaler nel cluster e verificarne il funzionamento.

- Il cluster Amazon EKS è già esistente. Se non lo è, consulta Nozioni di base.
- Kubernetes Metrics Server è già installato. Per ulteriori informazioni, consulta the section called "Metrics Server".
- Stai utilizzando un client `kubectl` che è configurato per comunicare con il cluster Amazon EKS.
- OpenSSL `1.1.1` o versioni successive deve essere installato sul dispositivo.

## Implementazione di Vertical Pod Autoscaler

In questa sezione, verrà implementato Vertical Pod Autoscaler nel cluster.

1. Aprire una finestra del terminale e passare a una directory in cui si desidera scaricare il codice sorgente di Vertical Pod Autoscaler.
2. Clona il kubernetes/autoscaler GitHub repository.

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. Passare alla directory `vertical-pod-autoscaler`.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (Facoltativo) Se è già stata implementata un'altra versione di Vertical Pod Autoscaler, rimuoverla con il comando seguente.

```
./hack/vpa-down.sh
```

5. Se i nodi non hanno accesso Internet al registro del container `registry.k8s.io`, devi estrarre le seguenti immagini e inviarle al tuo repository privato. Per ulteriori informazioni su come estrarre e inviare immagini al tuo repository privato, consulta la sezione the section called "Copia di un'immagine in un repository".

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

   Se invii le immagini a un repository Amazon ECR privato, sostituisci `registry.k8s.io` nei manifesti con il tuo registro. Sostituisci *`111122223333`* con l'ID del tuo account. *`region-code`*Sostituiscilo con la AWS regione in cui si trova il cluster. I comandi seguenti presuppongono

che tu abbia dato al repository lo stesso nome del repository nel manifesto. Se hai dato un nome diverso al repository, dovrai modificare anch'esso.

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/updater-deployment.yaml
```

6. Implementare Vertical Pod Autoscaler nel cluster mediante il comando seguente.

```
./hack/vpa-up.sh
```

7. Verifica che i pod di Vertical Pod Autoscaler siano stati creati correttamente.

```
kubectl get pods -n kube-system
```

Di seguito viene riportato un output di esempio.

```
NAME                                        READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

## Test dell'installazione di Vertical Pod Autoscaler

In questa sezione, verrà implementata un'applicazione di esempio per verificare il corretto funzionamento di Vertical Pod Autoscaler.

1. Implementare l'applicazione di esempio `hamster.yaml` per Vertical Pod Autoscaler mediante il comando seguente.

```
kubectl apply -f examples/hamster.yaml
```

2. Recupera i pod dall'applicazione esemplificativa `hamster`.

```
kubectl get pods -l app=hamster
```

Di seguito viene riportato un output di esempio.

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. Descrivi uno dei pod per visualizzare la prenotazione di `cpu` e `memory`. Sostituisci *`c7d89d6db-rglf5`* con uno degli ID restituiti nell'output del passaggio precedente.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

Di seguito viene riportato un output di esempio.

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

Nota che il pod originale riserva 100 millicpu di CPU e 50 mebibyte di memoria. Per questa applicazione di esempio, 100 millicpu sono inferiori a quelli necessari al Pod per funzionare, quindi

lo è. CPU-constrained Il pod riserva inoltre una quantità di memoria nettamente inferiore rispetto alla necessaria. L'implementazione di Vertical Pod Autoscaler `vpa-recommender` analizza i pod hamster per verificare se i requisiti di CPU e memoria siano appropriati. Se sono necessari adeguamenti, `vpa-updater` rilancia i pod con valori aggiornati.

4. Attendi il lancio `vpa-updater` di un nuovo Hamster Pod. Potrebbero essere necessari uno o due minuti. È possibile monitorare i pod con il comando seguente.

> **Note**
>
> Se non si è sicuri che sia stato avviato un nuovo pod, confronta i nomi dei pod facendo riferimento all'elenco precedente. All'avvio del nuovo pod, sarà visualizzato il relativo nome.

```
kubectl get --watch Pods -l app=hamster
```

5. Quando viene avviato un nuovo hamster Pod, descrivilo e visualizza le prenotazioni aggiornate di CPU e memoria.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

Di seguito viene riportato un output di esempio.

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
```

```
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
[...]
```

Nell'output precedente puoi osservare che la prenotazione della `cpu` è passata a 587 millicpu, ossia un valore di oltre cinque volte superiore a quello originale. La `memory` è aumentata a 262.144 KB (circa 250 mebibyte), ossia cinque volte il valore originale. Questo pod disponeva di un numero insufficiente di risorse e pertanto Vertical Pod Autoscaler ha corretto la stima con un valore più adeguato.

6. Descrivere la risorsa `hamster-vpa` per visualizzare la nuova raccomandazione.

```
kubectl describe vpa/hamster-vpa
```

Di seguito viene riportato un output di esempio:

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":{},"name":"hamster-
vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
```

```
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. Dopo aver provato l'applicazione di esempio, è possibile eliminarla con il comando seguente.

```
kubectl delete -f examples/hamster.yaml
```

# Scalare le implementazioni dei pod con Horizontal Pod Autoscaler

Horizontal Pod Autoscaler di Kubernetes scala automaticamente il numero di pod in un'implementazione, un controller di replica o un set di repliche in base all'utilizzo della CPU di tale risorsa. Ciò consente alle applicazioni di aumentare orizzontalmente le risorse in base all'aumento della domanda, o di ridurle orizzontalmente quando non sono necessarie, liberando così i nodi per altre applicazioni. Quando imposti una percentuale di utilizzo della CPU target, Horizontal Pod Autoscaler aumenta o riduce le risorse dell'applicazione in modo conforme all'obiettivo prefissato.

Horizontal Pod Autoscaler è una risorsa API standard in Kubernetes il cui funzionamento richiede solo l'installazione di una fonte di parametri, (ad esempio Kubernetes Metrics Server), nel cluster Amazon EKS. Non è necessario implementare o installare Horizontal Pod Autoscaler sul cluster per iniziare a dimensionare le applicazioni. Per ulteriori informazioni, consulta Horizontal Pod Autoscaler nella documentazione Kubernetes.

Utilizza questo argomento per preparare Horizontal Pod Autoscaler per il cluster Amazon EKS e per verificare che funzioni con un'applicazione di esempio.

> **Note**
>
> Questo argomento si basa su Horizontal Pod autoscaler walkthrough nella documentazione Kubernetes.

- Il cluster Amazon EKS è già esistente. Se non lo è, consulta Nozioni di base.
- Kubernetes Metrics Server è già installato. Per ulteriori informazioni, consulta the section called "Metrics Server".
- Stai utilizzando un client `kubectl` che è configurato per comunicare con il cluster Amazon EKS.

## Esecuzione di un'applicazione di test di Horizontal Pod Autoscaler

In questa sezione implementerai un'applicazione di esempio per verificare il corretto funzionamento di Horizontal Pod Autoscaler.

> **Note**
>
> Questo esempio si basa su Horizontal Pod autoscaler walkthrough nella documentazione Kubernetes.

1. Implementa un'applicazione server Web Apache con il comando seguente.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   Questo pod di server Web Apache ha un limite di CPU di 500 millicpu e serve sulla porta 80.
2. Crea una risorsa Horizontal Pod Autoscaler per l'implementazione di `php-apache`.

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   Questo comando crea un componente di scalabilità automatica che punta a un utilizzo della CPU del 50% per l'implementazione, con un minimo di un pod e un massimo di dieci pod. Quando il carico medio della CPU è inferiore al 50%, tale componente cerca di ridurre il

numero di pod nell'implementazione, fino a un minimo di uno. Quando il carico è maggiore del 50%, il componente di scalabilità automatica cerca di aumentare il numero di pod nell'implementazione, fino a un massimo di dieci. Per ulteriori informazioni, consulta How does a HorizontalPodAutoscaler work? nella documentazione Kubernetes.

3. Descrivere il componente di scalabilità automatica con il comando seguente per visualizzarne i dettagli.

```
kubectl get hpa
```

Di seguito viene riportato un output di esempio:

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
```

Come si può vedere, il carico attuale della CPU è `0%` perché non è ancora presente il carico sul server. Il conteggio dei pod è già al limite più basso (uno), quindi non può essere ridotto.

4. Crea un carico per il server Web eseguendo un container.

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. Per osservare l'incremento dell'implementazione, esegui periodicamente il seguente comando in un terminale separato dal terminale in cui hai eseguito il passaggio precedente.

```
kubectl get hpa php-apache
```

Di seguito viene riportato un output di esempio:

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5          4m44s
```

Potrebbe essere necessario più di un minuto per aumentare il numero di repliche. Finché la percentuale effettiva della CPU è superiore alla percentuale di destinazione, il conteggio delle repliche aumenta fino a 10. In questo caso, è `250%`, quindi il numero di `REPLICAS` continua ad aumentare.

> **Note**
>
> Potrebbero essere necessari alcuni minuti prima che il numero di repliche raggiunga il limite massimo. Se, ad esempio, sono necessarie solo 6 repliche affinché il carico della CPU rimanga pari o inferiore al 50%, il carico non verrà dimensionato oltre 6 repliche.

6. Arresta il carico. Nella finestra del terminale in cui si sta generando il carico, arrestare quest'ultimo tenendo premuti i tasti `Ctrl+C`. È possibile guardare il dimensionamento delle repliche nuovamente a 1 eseguendo di nuovo il seguente comando nel terminale corrispondente.

```
kubectl get hpa
```

   Di seguito viene riportato un output di esempio:

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          25m
```

> **Note**
>
> Il periodo di tempo di default per il ridimensionamento è di cinque minuti, quindi ci vorrà un po' di tempo prima di vedere il conteggio delle repliche raggiungere di nuovo 1, anche quando la percentuale corrente della CPU è 0. L'intervallo di tempo è modificabile.
> Per ulteriori informazioni, consulta Horizontal Pod Autoscaler nella documentazione Kubernetes.

7. Al termine dell'esercitazione con l'applicazione di esempio, elimina le risorse `php-apache`.

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

# Esecuzione del routing del traffico TCP e UDP con Network Load Balancer

> **Note**
>
> Novità: la modalità automatica di Amazon EKS automatizza le attività di routine per il bilanciamento del carico. Per ulteriori informazioni, consulta:
>
> - the section called "Implementa il bilanciatore del carico"
> - the section called "Creare servizio"

Il traffico di rete è bilanciato al carico `L4` del modello OSI. Per bilanciare il carico del traffico delle applicazioni`L7`, distribuisci un Kubernetes`ingress`, che fornisce un Application Load AWS Balancer. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di applicazione". Per ulteriori informazioni sulle differenze tra i due tipi di bilanciamento del carico, consulta le funzionalità di Elastic Load Balancing sul AWS sito Web.

Quando crei un Kubernetes `Service` di tipo Kubernetes`LoadBalancer`, il controller di bilanciamento del carico del provider AWS cloud crea AWSClassic Load Balancer per impostazione predefinita, ma può anche creare Network Load Balancer. AWS Questo controller in futuro riceverà solo correzioni di bug critici. Per ulteriori informazioni sull'utilizzo del controller di bilanciamento del carico del provider AWS cloud, consulta il controller di bilanciamento del carico del provider AWS cloud nella documentazione di Kubernetes. Il suo utilizzo non è descritto in questo argomento.

Ti consigliamo di utilizzare una versione `2.7.2` o successiva del AWS Load Balancer Controller anziché il controller di bilanciamento del carico del provider AWS cloud. Il AWS Load Balancer Controller crea AWS Network Load Balancer, ma non crea AWS Classic Load Balancer. Il resto di questo argomento riguarda l'utilizzo del AWS Load Balancer Controller.

Un AWS Network Load Balancer può bilanciare il carico del traffico di rete verso i Pod distribuiti su destinazioni IP e istanze di Amazon EC2, verso destinazioni IP AWS Fargate o su nodi ibridi Amazon EKS come destinazioni IP. Per ulteriori informazioni, vedere AWS Load Balancer Controller su. GitHub

# Prerequisiti

Prima di poter bilanciare il carico del traffico di rete utilizzando il AWS Load Balancer Controller, è necessario soddisfare i seguenti requisiti.

- Avere un cluster esistente. Se non si dispone di un cluster esistente, consultare Nozioni di base. Se è necessario aggiornare la versione di un cluster esistente, vedere the section called "Aggiornamento della versione di Kubernetes".
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.
- Disporre di almeno una sottorete. Se si trovano più sottoreti con tag in una zona di disponibilità, il controller sceglie la sottorete il cui ID sottorete viene prima in ordine lessicografico. La sottorete deve avere a disposizione almeno otto indirizzi IP.
- Se utilizzi la versione AWS Load Balancer Controller `2.1.1` o precedente, le sottoreti devono essere etichettate come segue. Se si utilizza la versione `2.1.2` o successiva, questo tag è facoltativo. Potresti voler etichettare una sottorete se hai più cluster in esecuzione nello stesso VPC o più AWS servizi che condividono sottoreti in un VPC e desideri un maggiore controllo su dove vengono forniti i load balancer per ogni cluster. Se specifichi esplicitamente gli ID di sottorete come annotazione su un oggetto servizio, Kubernetes e il Load AWS Balancer Controller utilizzano tali sottoreti direttamente per creare il load balancer. L'assegnazione di tag delle sottoreti non è obbligatoria se si sceglie di utilizzare questo metodo per il provisioning dei bilanciatori del carico ed è possibile ignorare i seguenti requisiti di assegnazione di tag delle sottoreti pubbliche e private. Sostituisci *`my-cluster`* con il nome del cluster.
  - Chiave: `kubernetes.io/cluster/<my-cluster>`
  - Valore: `shared` o `owned`
- Le sottoreti pubbliche e private devono soddisfare i seguenti requisiti, a meno che non si specifichino esplicitamente gli ID sottorete come annotazione in un oggetto di servizio o di ingresso. Se esegui il provisioning dei load balancer specificando esplicitamente gli ID di sottorete come annotazione su un servizio o un oggetto di ingresso, Kubernetes e il Load Balancer Controller utilizzano tali sottoreti direttamente per creare il AWS load balancer e i seguenti tag non sono necessari.
  - Sottoreti private: deve essere taggato nel seguente formato. In questo modo Kubernetes e Load Balancer Controller sanno che le sottoreti possono essere utilizzate per bilanciatori di AWS carico interni. Se utilizzi `eksctl` o un AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato al momento

della creazione. Per ulteriori informazioni sui modelli AWS CloudFormation VPC di Amazon EKS, consulta. the section called "Crea un VPC"

- Chiave: `kubernetes.io/role/internal-elb`
- Valore: 1

- Sottoreti pubbliche – Deve essere taggato nel seguente formato. In questo modo, Kubernetes utilizza solo tali sottoreti per load balancer esterni, anziché scegliere una sottorete pubblica in ciascuna zona di disponibilità (in ordine lessicografico per ID sottorete). Se utilizzi `eksctl` o un AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato al momento della creazione. Per ulteriori informazioni sui modelli AWS CloudFormation VPC di Amazon EKS, consulta. the section called "Crea un VPC"
  - Chiave: `kubernetes.io/role/elb`
  - Valore: 1

Se i tag del ruolo della sottorete non vengono aggiunti in modo esplicito, il controller del servizio Kubernetes esamina la tabella di routing delle sottoreti VPC del cluster per determinare se la sottorete è privata o pubblica. Si consiglia di non fare affidamento su questo comportamento e di aggiungere esplicitamente i tag di ruolo pubblici o privati. Il AWS Load Balancer Controller non esamina le tabelle delle rotte e richiede la presenza dei tag privati e pubblici per una corretta individuazione automatica.

## Considerazioni

- La configurazione del load balancer è controllata da annotazioni che vengono aggiunte al manifest del servizio. Le annotazioni del servizio sono diverse quando si utilizza il AWS Load Balancer Controller rispetto a quando si utilizza AWS il controller di bilanciamento del carico del provider cloud. Assicurati di controllare le annotazioni per il AWS Load Balancer Controller prima di distribuire i servizi.
- Quando si utilizza il plug-in Amazon VPC CNI per Kubernetes, il Load AWS Balancer Controller può bilanciare il carico su destinazioni IP o istanze di Amazon EC2 e destinazioni IP Fargate. Quando utilizzi plug-in CNI compatibili alternativi, il controller può bilanciare il carico solo per le destinazioni di istanza, a meno che non si stia bilanciando il carico su Amazon EKS Hybrid Nodes. Per i nodi ibridi, il controller può bilanciare il carico delle destinazioni IP. Per ulteriori informazioni sui tipi di destinazioni dei Network Load Balancer, consultare Target type (Tipo di destinazione) nella Guida per l'utente di Network Load Balancer.

- Se si desidera aggiungere tag al bilanciatore del carico quando o dopo la sua creazione, aggiungere la seguente annotazione nella specifica del servizio. Per ulteriori informazioni, consulta AWS Resource Tags nella documentazione del AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- È possibile assegnare Indirizzi IP elastici al Network Load Balancer aggiungendo la seguente annotazione. Sostituisci i valori `Allocation IDs` di esempio con i tuoi indirizzi IP elastici. Il numero di `Allocation IDs` deve corrispondere al numero di sottoreti utilizzate per il load balancer. Per ulteriori informazioni, consultare la documentazione sul Controller del load balancer AWS.

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations: eipalloc-
xxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- Amazon EKS aggiunge una regola in entrata al gruppo di sicurezza del nodo per il traffico client e una regola per ogni sottorete del bilanciatore del carico nel VPC per i controlli dell'integrità per ogni Network Load Balancer creato. L'implementazione di un servizio di tipo `LoadBalancer` può non riuscire se Amazon EKS tenta di creare regole che superano la quota per il numero massimo di regole consentite per un gruppo di sicurezza. Per ulteriori informazioni, consultare Gruppi di sicurezza in Quote di Amazon VPC nella Guida per l'utente di Amazon VPC. Considerare le opzioni seguenti per ridurre al minimo le possibilità di superare il numero massimo di regole per un gruppo di sicurezza:
  - Richiedere un aumento delle regole per ogni quota di gruppo di sicurezza. Per ulteriori informazioni, consultare Richiesta di un aumento di quota nella Guida per l'utente di Service Quotas.
  - Al posto di destinazioni di istanza, utilizzare destinazioni IP. Con le destinazioni IP, è possibile condividere le regole per le stesse porte di destinazione. È possibile specificare manualmente le sottoreti del load balancer con un'annotazione. Per ulteriori informazioni, consulta Annotations on GitHub.
  - Utilizzare un ingresso invece di un servizio di tipo `LoadBalancer` per inviare traffico al tuo servizio. L' AWS Application Load Balancer richiede meno regole rispetto ai Network Load Balancer. È possibile condividere un ALB su più ingressi. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di applicazione". Non è possibile condividere un Network Load Balancer su più servizi.
  - Implementa i cluster su più account.

- Se i pod vengono eseguiti su Windows in un cluster Amazon EKS, un singolo servizio con un bilanciatore del carico può supportare fino a 1024 pod back-end. Ogni pod ha il proprio indirizzo IP univoco.
- Consigliamo di creare nuovi Network Load Balancer solo con il Load AWS Balancer Controller. Il tentativo di sostituire i Network Load Balancer esistenti creati con il controller di bilanciamento del carico del provider di servizi AWS cloud può comportare la creazione di più Network Load Balancer che potrebbero causare tempi di inattività delle applicazioni.

## Creazione di un Network Load Balancer

È possibile creare un Network Load Balancer con destinazioni IP o istanza.

### Creare un Network Load Balancer: destinazioni IP

- È possibile utilizzare destinazioni IP con pod implementati su nodi Amazon EC2, Fargate o Amazon EKS Hybrid Nodes. Il servizio Kubernetes deve essere creato come tipo `LoadBalancer`. Per ulteriori informazioni, consulta Type nella documentazione di Kubernetes. LoadBalancer

  Per creare un load balancer che utilizzi destinazioni IP, aggiungere le seguenti annotazioni a un manifesto del servizio e implementare il servizio. Il `external` valore per `aws-load-balancer-type` è ciò che fa sì che il AWS Load Balancer Controller, anziché il controller del load balancer del provider AWS cloud, crei il Network Load Balancer. É possibile visualizzare un manifesto del servizio di esempio con le annotazioni.

  ```
  service.beta.kubernetes.io/aws-load-balancer-type: "external"
  service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
  ```

  > **Note**
  >
  > Se si sta eseguendo il bilanciamento del carico sui pod `IPv6`, aggiungere la seguente annotazione. È possibile bilanciare il carico solo su destinazioni da `IPv6` a IP, non su destinazioni di istanza. Senza questa annotazione, il bilanciamento del carico è su `IPv4`.

  ```
  service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
  ```

Per impostazione predefinita, i Network Load Balancer vengono creati con `internal` `aws-load-balancer-scheme`. È possibile avviare i Network Load Balancer in qualsiasi sottorete del VPC del cluster, incluse le sottoreti non specificate al momento della creazione del cluster.

Kubernetes esamina la tabella di routing delle sottoreti per stabilire se sono pubbliche o private. Le sottoreti pubbliche hanno una route direttamente a Internet tramite un gateway Internet; non vale altrettanto per le sottoreti private.

Se si desidera creare un Network Load Balancer in una sottorete pubblica per il bilanciamento del carico sui nodi Amazon EC2 (Fargate può essere solo privato), specificare `internet-facing` con la seguente annotazione:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> Al fine di garantire la compatibilità con le versioni precedenti, l'annotazione `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` è ancora supportata. Consigliamo, tuttavia, di utilizzare le annotazioni precedenti per i load balancer anziché `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`.

> **Important**
>
> Non modificare le annotazioni dopo aver creato il servizio. Se è necessario modificarlo, eliminare l'oggetto di servizio e crearlo nuovamente con il valore desiderato per questa annotazione.

## Creare un Network Load Balancer: destinazioni di istanza

- Il controller di bilanciamento del carico del provider AWS cloud crea Network Load Balancer solo con destinazioni di istanza. La versione `2.2.0` e le successive del AWS Load Balancer Controller creano anche Network Load Balancer con destinazioni di istanza. Ti consigliamo di utilizzarlo, anziché il controller di bilanciamento del carico del provider AWS cloud, per creare nuovi Network Load Balancer. È possibile utilizzare le destinazioni di istanza di Network Load Balancer con pod

implementati sui nodi Amazon EC2, ma non su Fargate. Per bilanciare il carico del traffico di rete tra i pod implementati su Fargate, è necessario utilizzare destinazioni IP.

Per implementare un Network Load Balancer in una sottorete privata, la specifica del servizio deve avere le seguenti annotazioni. É possibile visualizzare un manifesto del servizio di esempio con le annotazioni. Il `external` valore per `aws-load-balancer-type` è ciò che fa sì che il AWS Load Balancer Controller, anziché il controller del load balancer del provider AWS cloud, crei il Network Load Balancer.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

Per impostazione predefinita, i Network Load Balancer vengono creati con `internal` `aws-load-balancer-scheme`. Per i Network Load Balancer interni, il cluster Amazon EKS deve essere configurato per l'utilizzo di almeno una sottorete privata nel VPC. Kubernetes esamina la tabella di routing delle sottoreti per stabilire se sono pubbliche o private. Le sottoreti pubbliche hanno una route direttamente a Internet tramite un gateway Internet; non vale altrettanto per le sottoreti private.

Se desideri creare un Network Load Balancer in una sottorete pubblica per bilanciare il carico sui nodi Amazon EC2, `internet-facing` specificalo con la seguente annotazione:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Important**
>
> Non modificare le annotazioni dopo aver creato il servizio. Se è necessario modificarlo, eliminare l'oggetto di servizio e crearlo nuovamente con il valore desiderato per questa annotazione.

## (Facoltativo) Implementare un'applicazione di esempio

- Almeno una sottorete pubblica o privata nel VPC del cluster.
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.

1. Se si sta eseguendo l'implementazione su Fargate, assicurarsi di avere una sottorete privata disponibile nel VPC e di creare un profilo Fargate. Se non si sta eseguendo l'implementazione su Fargate, salta questo passaggio. Puoi creare il profilo eseguendo il seguente comando o nella Console di gestione AWS utilizzando gli stessi valori per `name` e `namespace` che si trovano nel comando. Sostituire i valori di esempio con i propri valori.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name nlb-sample-app \
    --namespace nlb-sample-app
```

2. Implementa un'applicazione di esempio.

   a. Crea un namespace per l'applicazione.

```
kubectl create namespace nlb-sample-app
```

   b. Salva nel computer i seguenti contenuti in un file denominato `sample-deployment.yaml`.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nlb-sample-app
  namespace: nlb-sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

   c. Applica il file manifesto al cluster.

```
kubectl apply -f sample-deployment.yaml
```

3. Crea un servizio con un Network Load Balancer connesso a Internet che bilanci il carico sulle destinazioni IP.

   a. Salva sul computer i seguenti contenuti in un file denominato `sample-service.yaml`. Se si sta eseguendo l'implementazione sui nodi Fargate, rimuovere la riga `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing`.

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

   b. Applica il file manifesto al cluster.

```
kubectl apply -f sample-service.yaml
```

4. Verifica che il servizio sia stato distribuito.

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

   Di seguito viene riportato un output di esempio.

```
NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                 PORT(S)        AGE
sample-service   LoadBalancer   10.100.240.137   k8s-nlbsampl-nlbsampl-xxxxxxxxxx-
xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com  80:32400/TCP   16h
```

> **Note**
>
> I valori di *10.100.240.137* e *xxxxxxxxxx* - *xxxxxxxxxxxxxxxx* saranno diversi dall'output di esempio (saranno univoci per il sistema di bilanciamento del carico) e *us-west-2* potrebbero essere diversi per te, a seconda della AWS regione in cui si trova il cluster.

5. Apri Amazon EC2 Console di gestione AWS. Seleziona Target Groups (Gruppi di destinazioni) in Load Balancing (Bilanciamento del carico) nel pannello di navigazione a sinistra. Nella colonna Nome, seleziona il nome del gruppo di destinazione in cui il valore nella colonna bilanciatore del carico corrisponde al nome nella colonna `EXTERNAL-IP` dell'output nel passaggio precedente. Ad esempio, se l'output era uguale all'output indicato sopra sarà necessario selezionare il gruppo di destinazione denominato `k8s-default-samplese-`*xxxxxxxxxx* . Target type (Tipo di destinazione) è `IP` perché è stato specificato nel manifesto del servizio di esempio.
6. Selezionare il proprio Gruppo di destinazione e poi scegliere la scheda Destinazioni. In Destinazioni registrate, verranno visualizzati tre indirizzi IP delle tre repliche implementate in un passaggio precedente. Prima di continuare, attendere fino a quando lo stato di tutti gli obiettivi è Integro. Potrebbero passare diversi minuti prima che tutti gli obiettivi siano `healthy`. Gli obiettivi potrebbero essere in uno stato `unhealthy` prima di passare a uno stato `healthy`.
7. Invia il traffico al servizio sostituendo *xxxxxxxxxx-xxxxxxxxxxxxxxxx* e *us-west-2* con i valori restituiti nell'output di un operazione precedente per `EXTERNAL-IP`. Se hai eseguito l'implementazione a una sottorete privata, devi visualizzare la pagina da un dispositivo all'interno del VPC, ad esempio un host bastione. Per ulteriori informazioni, consulta Bastion host Linux in AWS.

   ```
   curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
   ```

   Di seguito viene riportato un output di esempio.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```
8. Al termine dell'implementazione, del servizio e del namespace esemplificativi, rimuoverli.

```
kubectl delete namespace nlb-sample-app
```

# Instradare il traffico di applicazioni e HTTP con Application Load Balancer

> **Note**
>
> Novità: la modalità automatica di Amazon EKS automatizza le attività di routine per il bilanciamento del carico. Per ulteriori informazioni, consulta:
>
> - the section called "Implementa il bilanciatore del carico"
> - the section called "Creare una classe in ingresso"

Quando crei un Kubernetes`ingress`, viene fornito un Application Load AWS Balancer (ALB) che bilancia il carico del traffico delle applicazioni. Per ulteriori informazioni, consulta Cos'è un Application Load Balancer? nella Guida per l'utente di Application Load Balancer e Ingressi nella documentazione Kubernetes. Gli ALB possono essere utilizzati con i Pod distribuiti sui nodi o su Fargate. AWS È possibile implementare un ALB a sottoreti pubbliche o private.

Il traffico delle applicazioni è bilanciato a `L7` del modello OSI. Per bilanciare il carico del traffico di rete a `L4`, implementare un `service` Kubernetes del tipo `LoadBalancer`. Questo tipo fornisce un AWS Network Load Balancer. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di rete". Per ulteriori informazioni sulle differenze tra i due tipi di bilanciamento del carico, consulta le funzionalità di Elastic Load Balancing sul AWS sito Web.

## Prerequisiti

Prima di poter bilanciare il carico del traffico dell'applicazione in un'applicazione, è necessario soddisfare i seguenti requisiti.

- Avere un cluster esistente. Se non si dispone di un cluster esistente, consultare Nozioni di base. Se è necessario aggiornare la versione di un cluster esistente, vedere the section called "Aggiornamento della versione di Kubernetes".
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.

- Almeno due sottoreti devono trovarsi in diverse zone di disponibilità. Il AWS Load Balancer Controller sceglie una sottorete da ogni zona di disponibilità. Quando si trovano più sottoreti taggate in una zona di disponibilità, il controller sceglie la sottorete il cui ID sottorete viene prima in ordine lessicografico. Ogni sottorete deve avere a disposizione almeno otto indirizzi IP.

  Se utilizzi più gruppi di sicurezza collegati ai nodi di lavoro, esattamente un gruppo di sicurezza deve essere etichettato come segue. Sostituisci *my-cluster* con il nome del cluster.

  - Chiave: `kubernetes.io/cluster/<my-cluster>`
  - Valore: `shared` o `owned`
- Se utilizzi la versione AWS Load Balancer Controller `2.1.1` o precedente, le sottoreti devono essere etichettate nel formato seguente. Se utilizzi la versione `2.1.2` o successiva, l'assegnazione di tag è facoltativa. Tuttavia, si consiglia di taggare una sottorete nel caso si verifichi una delle seguenti situazioni. Hai più cluster in esecuzione nello stesso VPC o AWS più servizi che condividono sottoreti in un VPC. Oppure, si desidera un maggiore controllo sulla posizione in cui viene eseguito il provisioning dei bilanciatori di carico per ciascun cluster. Sostituisci *my-cluster* con il nome del cluster.
  - Chiave: `kubernetes.io/cluster/<my-cluster>`
  - Valore – `shared` o `owned`
- Le sottoreti pubbliche e private devono soddisfare i seguenti requisiti. Ciò avviene a condizione che non si indichino esplicitamente gli ID sottorete come annotazione in un oggetto di servizio o di ingresso. Si supponga di effettuare il provisioning dei load balancer specificando esplicitamente gli ID sottorete come annotazione in un oggetto di servizio o di ingresso. In questa situazione, Kubernetes e il controller del AWS bilanciamento del carico utilizzano direttamente tali sottoreti per creare il sistema di bilanciamento del carico e i seguenti tag non sono necessari.
  - Sottoreti private: deve essere taggato nel seguente formato. In questo modo Kubernetes e il controller del bilanciamento del carico sanno che le sottoreti possono essere utilizzate per bilanciatori di AWS carico interni. Se utilizzi `eksctl` o un AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato al momento della creazione. Per ulteriori informazioni sui modelli AWS CloudFormation VPC di Amazon EKS, consulta. the section called "Crea un VPC"
    - Chiave: `kubernetes.io/role/internal-elb`
    - Valore: 1
  - Sottoreti pubbliche – Deve essere taggato nel seguente formato. In questo modo Kubernetes sa di utilizzare solo le sottoreti specificate per i load balancer esterni. In questo modo, Kubernetes non sceglie una sottorete pubblica in ogni zona di disponibilità (in base all'ordine lessicografico

degli ID sottorete). Se utilizzi `eksctl` o un AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato al momento della creazione. Per ulteriori informazioni sui modelli AWS CloudFormation VPC di Amazon EKS, consulta. the section called "Crea un VPC"

- Chiave: `kubernetes.io/role/elb`
- Valore: 1

Se i tag del ruolo della sottorete non vengono aggiunti in modo esplicito, il controller del servizio Kubernetes esamina la tabella di routing delle sottoreti VPC del cluster. Ciò consente di determinare se la sottorete è privata o pubblica. Si consiglia di non fare affidamento su questo comportamento. È preferibile, invece, aggiungere esplicitamente i tag del ruolo privato o pubblico. Il AWS Load Balancer Controller non esamina le tabelle delle rotte. Richiede inoltre la presenza di tag privati e pubblici per un efficiente rilevamento automatico.

- Il AWS Load Balancer Controller crea ALB e le AWS risorse di supporto necessarie ogni volta che viene creata una risorsa di ingresso Kubernetes nel cluster con l'annotazione. `kubernetes.io/ingress.class: alb` La risorsa di ingresso configura l'ALB per instradare il traffico HTTP o HTTPS verso pod differenti all'interno del cluster. Per assicurarti che i tuoi oggetti in ingresso utilizzino il AWS Load Balancer Controller, aggiungi la seguente annotazione alla specifica di ingresso di Kubernetes. Per ulteriori informazioni, consulta Specifiche di ingresso su GitHub.

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> Se bilanci il carico sui pod `IPv6`, aggiungi la seguente annotazione alla specifica di ingresso. È possibile bilanciare il carico solo su destinazioni da `IPv6` a IP, non su destinazioni di istanza. Senza questa annotazione, il bilanciamento del carico è su `IPv4`.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- Il AWS Load Balancer Controller supporta le seguenti modalità di traffico:
  - Istanza: registra i nodi all'interno del cluster come destinazioni per l'ALB. Il traffico che raggiunge l'ALB viene indirizzato a `NodePort` per il servizio e quindi inoltrato ai pod. Questa è la

modalità di traffico predefinita. È anche possibile specificarla esplicitamente con l'annotazione `alb.ingress.kubernetes.io/target-type: instance`.

> **Note**
>
> Il servizio Kubernetes deve specificare il tipo `NodePort` o `LoadBalancer` per utilizzare questa modalità di traffico.

- IP – Registra i pod come destinazioni per l'ALB. Il traffico che raggiunge l'ALB viene indirizzato direttamente ai pod per il servizio. È necessario specificare l'annotazione `alb.ingress.kubernetes.io/target-type: ip` per utilizzare questa modalità di traffico. Il tipo di destinazione IP è richiesto quando i pod di destinazione sono in esecuzione su Fargate o Amazon EKS Hybrid Nodes.

- Per assegnare i tag agli ALB creati dal controller, aggiungere la seguente annotazione al controller: `alb.ingress.kubernetes.io/tags`. Per un elenco di tutte le annotazioni disponibili supportate dal Load AWS Balancer Controller, consulta Ingress annotations on. GitHub
- L'aggiornamento o il downgrade della versione del controller ALB può introdurre modifiche importanti alle funzionalità che si basano su di essa. Per ulteriori informazioni sulle modifiche che vengono introdotte in ogni versione, consulta le note di rilascio del controller ALB su GitHub.

## Riutilizzare ALB con gruppi di ingresso

Puoi condividere un Application Load Balancer su più risorse di servizio tramite `IngressGroups`.

Per aggiungere un ingresso a un gruppo, aggiungere la seguente annotazione a una specifica di risorsa di ingresso Kubernetes.

```
alb.ingress.kubernetes.io/group.name: my-group
```

Il nome del gruppo deve:

- Avere una lunghezza pari o inferiore a 63 caratteri.
- Essere formato da lettere minuscole, numeri, `-`, e `.`
- Iniziare e finire con una lettera o un numero.

Il controller unisce automaticamente le regole di ingresso per tutti gli ingressi nello stesso gruppo di ingresso. Le supporta con un singolo ALB. La maggior parte delle annotazioni definite in un ingresso

si applica solo ai percorsi definiti da tale ingresso. Per impostazione predefinita, le risorse in ingresso non appartengono ad alcun gruppo di ingresso.

> **Warning**
>
> Potenziale rischio per la sicurezza
> Specifica un gruppo di ingressi per un ingresso solo quando tutti gli utenti Kubernetes che dispongono dell'autorizzazione RBAC per creare o modificare le risorse di ingresso rientrano nello stesso limite di attendibilità. Se si aggiunge l'annotazione con un nome di gruppo, altri utenti Kubernetes potrebbero creare o modificare i propri ingressi così da appartenere allo stesso gruppo di ingressi. Ciò può causare comportamenti indesiderati, ad esempio la sovrascrittura delle regole esistenti con regole con priorità più alta.

Puoi aggiungere un numero d'ordine della risorsa di ingresso.

```
alb.ingress.kubernetes.io/group.order: '10'
```

Il numero può variare tra 1 e 1.000. Viene considerato prima il numero più basso per tutti gli ingressi nello stesso gruppo di ingressi. Tutti gli ingressi senza questa annotazione vengono valutati con un valore pari a zero. La duplicazione delle regole con un numero più alto può causare la sovrascrittura delle regole con un numero più basso. Per impostazione predefinita, l'ordine delle regole tra gli ingressi all'interno dello stesso gruppo di ingresso viene determinato lessicograficamente in base allo spazio dei nomi e al nome.

> **Important**
>
> Assicurarsi che ogni ingresso nello stesso gruppo di ingressi disponga di un numero di priorità univoco. Non puoi avere numeri d'ordine duplicati tra gli ingressi.

## (Facoltativo) Implementare un'applicazione di esempio

- Almeno una sottorete pubblica o privata nel VPC del cluster.
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.

È possibile eseguire l'applicazione di esempio in un cluster che abbia nodi Amazon EC2, pod Fargate o entrambi.

1. Se non stai eseguendo l'implementazione su Fargate, salta questa fase. Se stai eseguendo l'implementazione su Fargate, crea un profilo Fargate. Puoi creare il profilo eseguendo il seguente comando o nella Console di gestione AWS utilizzando gli stessi valori per `name` e `namespace` che si trovano nel comando. Sostituire i valori di esempio con i propri valori.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name alb-sample-app \
    --namespace game-2048
```

2. Implementa il gioco 2048 come applicazione di esempio per verificare che il Load AWS Balancer Controller crei AWS un ALB come risultato dell'oggetto di ingresso. Completa i passaggi per il tipo di sottorete verso la quale stai eseguendo l'implementazione.

   a. Se stai eseguendo l'implementazione su pod in un cluster creato con la famiglia `IPv6`, passa alla fase successiva.

   - Pubblica:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

   - Privata:

     A. Eseguire il download del manifesto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

     B. Modificare il file e individuare la riga `alb.ingress.kubernetes.io/scheme: internet-facing`.

     C. Modificare *`internet-facing`* in `internal` e salvare il file.

     D. Applica il file manifesto al cluster.

```
kubectl apply -f 2048_full.yaml
```

b. Se stai eseguendo l'implementazione su pod in un cluster creato con la famiglia IPv6, completa la seguente procedura.

i. Eseguire il download del manifesto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

ii. Aprire il file in un editor e aggiungere la seguente riga alle annotazioni nella specifica di ingresso.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

iii. Se stai bilanciando il carico sui pod interni piuttosto che sui pod rivolti a Internet, modifica la riga che indica `alb.ingress.kubernetes.io/scheme: internet-facing` in `alb.ingress.kubernetes.io/scheme: internal`

iv. Salvare il file.

v. Applica il file manifesto al cluster.

```
kubectl apply -f 2048_full.yaml
```

3. Dopo pochi minuti, verificare che la risorsa di ingresso sia stata creata con il comando seguente.

```
kubectl get ingress/ingress-2048 -n game-2048
```

Di seguito viene riportato un output di esempio:

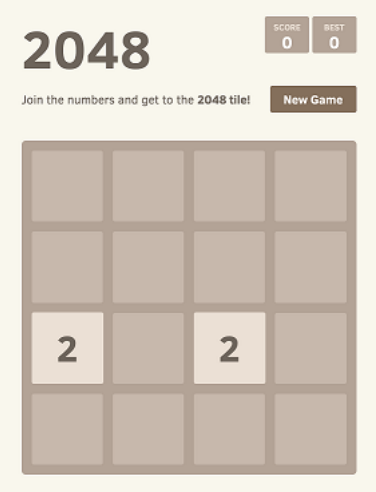
```
NAME           CLASS    HOSTS   ADDRESS
                      PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> Se hai creato il load balancer in una sottorete privata, il valore in `ADDRESS` nell'output precedente è preceduto da `internal-`.

Se l'ingresso non è stato creato correttamente dopo alcuni minuti, esegui il comando seguente per visualizzare i log del AWS Load Balancer Controller. Questi registri possono contenere messaggi di errore che consentono di diagnosticare eventuali problemi relativi all'implementazione.

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-balancer-
controller
```

1. Se è stata eseguita l'implementazione a una sottorete pubblica, aprire un browser e accedere all'URL ADDRESS dall'output del comando precedente per visualizzare l'applicazione di esempio. Se non visualizzi nulla, aggiorna il browser e riprova. Se hai eseguito l'implementazione a una sottorete privata, devi visualizzare la pagina da un dispositivo all'interno del VPC, ad esempio un host bastione. Per ulteriori informazioni, consulta Bastion host Linux in AWS.




2. Al termine della sperimentazione con l'applicazione di esempio, eliminala eseguendo uno dei seguenti comandi.
   - Se è stato applicato il manifesto anziché una copia scaricata, utilizzare il comando seguente.

     ```
     kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-
     balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
     ```

   - Se è stato il manifesto scaricato e modificato, utilizzare il seguente comando.

     ```
     kubectl delete -f 2048_full.yaml
     ```

## Limitazione degli indirizzi IP esterni assegnabili ai servizi

I servizi Kubernetes possono essere raggiunti dall'interno di un cluster tramite:

- Un indirizzo IP del cluster assegnato automaticamente da Kubernetes
- Qualsiasi indirizzo IP specificato per la proprietà `externalIPs` in una specifica di servizio. Gli indirizzi IP esterni non sono gestiti da Kubernetes e sono responsabilità dell'amministratore del cluster. Gli Indirizzi IP esterni specificati con `externalIPs` sono diversi dall'indirizzo IP esterno assegnato a un servizio di tipo `LoadBalancer` da un provider cloud.

Per ulteriori informazioni sui servizi Kubernetes, consulta Servizio nella documentazione Kubernetes. È possibile limitare gli indirizzi IP che possono essere specificati per `externalIPs` in una specifica di servizio.

1. Implementazione di `cert-manager` per gestire i certificati webhook. Per ulteriori informazioni, consultare la documentazione Cert-Manager.

   ```
   kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/
   cert-manager.yaml
   ```

2. Verifica che i pod `cert-manager` siano in esecuzione.

   ```
   kubectl get pods -n cert-manager
   ```

   Di seguito viene riportato un output di esempio.

   ```
   NAME                                       READY   STATUS    RESTARTS   AGE
   cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
   cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
   cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
   ```

3. Esamina i servizi esistenti per assicurarsi che a nessuno di essi siano assegnati indirizzi IP esterni non contenuti nell'intervallo CIDR per cui si desidera limitare gli indirizzi.

   ```
   kubectl get services -A
   ```

   Di seguito viene riportato un output di esempio.

   ```
   NAMESPACE                   NAME                                    TYPE
    CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
   cert-manager                cert-manager                            ClusterIP
    10.100.102.137   <none>          9402/TCP        20m
   ```

```
cert-manager                    cert-manager-webhook                    ClusterIP
 10.100.6.136     <none>           443/TCP         20m
default                         kubernetes                              ClusterIP
 10.100.0.1       <none>           443/TCP         2d1h
externalip-validation-system    externalip-validation-webhook-service   ClusterIP
 10.100.234.179   <none>           443/TCP         16s
kube-system                     kube-dns                                ClusterIP
 10.100.0.10      <none>           53/UDP,53/TCP   2d1h
my-namespace                    my-service                              ClusterIP
 10.100.128.10    192.168.1.1      80/TCP          149m
```

Se uno qualsiasi dei valori costituiscono indirizzi IP che non si trovano all'interno del blocco per cui si desidera limitare l'accesso, sarà necessario modificare gli indirizzi in modo che si trovino all'interno del blocco e implementare nuovamente i servizi. Ad esempio, al servizio `my-service` nell'output precedente è assegnato un indirizzo IP esterno che non rientra nell'intervallo CIDR esemplificativo del passaggio 5.

4. Eseguire il download del manifesto del webhook IP esterno. Puoi visualizzare il codice sorgente per il webhook anche su GitHub.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. Specifica i blocchi CIDR. Aprire il file scaricato nell'editor e rimuovere il `\#` all'inizio delle righe seguenti.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

Sostituire `10.0.0.0/8` con il proprio blocco CIDR. É possibile specificare tutti i blocchi che si desiderano. Se specifichi più blocchi, aggiungi una virgola tra i blocchi.

6. Se il cluster non si trova nella `us-west-2` AWS regione `us-west-2602401143452`, sostituisci e `amazonaws.com` nel file con i seguenti comandi. Prima di eseguire i comandi, sostituisci *`region-code`* e *`111122223333`* con il valore per la tua AWS regione dall'elenco in Visualizza i registri delle immagini dei contenitori Amazon per i componenti aggiuntivi Amazon EKS.

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. Applica il file manifesto al cluster.

```
kubectl apply -f externalip-webhook.yaml
```

Un tentativo di implementare un servizio nel cluster con un indirizzo IP specificato `externalIPs` che non è contenuto nei blocchi specificati nel passaggio Specificare i blocchi CIDR avrà esito negativo.

## Copia di un'immagine di container da un repository a un altro

In questo argomento viene descritto come estrarre un'immagine di container da un repository a cui i nodi non hanno accesso e inviarla a un repository a cui i nodi hanno accesso. Puoi inviare l'immagine ad Amazon ECR o a un repository alternativo al quale i tuoi nodi hanno accesso.

- Il motore Docker deve essere installato e configurato sul computer. Per ulteriori informazioni, consulta la sezione Installazione del motore Docker nella documentazione di Docker.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Un endpoint VPC di interfaccia per Amazon ECR se si desidera che i nodi estraggano o trasferiscano le immagini di container in un repository Amazon ECR privato sulla rete di Amazon. Per ulteriori informazioni, consulta la sezione Creazione degli endpoint VPC per Amazon ECR nella Guida per l'utente di Amazon Elastic Container Registry.

Completa i passaggi seguenti per estrarre un'immagine del container da un repository e inviarla al tuo repository. Negli esempi riportati in questo argomento, viene estratta l'immagine per l'helper di parametri del plugin CNI di Amazon VPC per Kubernetes. Quando segui questi passaggi, assicurati di sostituire i valori di esempio con i tuoi valori.

1. Se non si dispone ancora di un repository Amazon ECR o di un altro repository, crearne uno a cui i nodi abbiano accesso. Il seguente comando crea un repository privato Amazon ECR. Il nome di

un repository privato Amazon ECR deve iniziare con una lettera. Può contenere soltanto lettere minuscole, numeri, trattini (-), trattini bassi (_) e barre (/). Per ulteriori informazioni, consulta la sezione Creazione di un repository privato nella Guida per l'utente di Amazon Elastic Container Registry.

Puoi sostituire *cni-metrics-helper* con un qualsiasi valore a tua scelta. Come best practice, crea un repository separato per ogni immagine. Lo consigliamo perché i tag immagine devono essere univoci all'interno di un repository. Sostituisci *region-code* con una AWS regione supportata da Amazon ECR.

```
aws ecr create-repository --region region-code --repository-name cni-metrics-helper
```

2. Determina il registro, il repository e il tag (facoltativo) dell'immagine che i nodi devono estrarre. Questa informazione è nel formato `registry/repository[:tag]`.

   Molti degli argomenti di Amazon EKS sull'installazione delle immagini richiedono l'applicazione di un file manifesto o l'installazione dell'immagine utilizzando un grafico Helm. Tuttavia, prima di applicare un file manifesto o installare un grafico Helm, visualizzare il contenuto del manifesto o del file `values.yaml` del grafico. In questo modo, puoi determinare il registro, il repository e il tag da estrarre.

   Ad esempio, è possibile trovare la seguente riga nel file manifesto dell'helper di parametri del plugin CNI di Amazon VPC per Kubernetes. Il registro è `602401143452.dkr.ecr.us-west-2.amazonaws.com`, che è un registro privato Amazon ECR. Il repository è `cni-metrics-helper`.

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
   ```

   Potresti visualizzate le seguenti varianti per la posizione dell'immagine:

   - Solo `repository-name:tag`. In questo caso, `docker.io` di solito è il registro, ma non è specificato poiché di default Kubernetes gli antepone un nome di repository se non viene specificato alcun registro.
   - `repository-name/repository-namespace/repository:tag`. il namespace del repository è facoltativo, ma talvolta viene specificato dal proprietario del repository per la classificazione delle immagini. Ad esempio, tutte le immagini Amazon EC2 nella galleria pubblica Amazon ECR usano il namespace `aws-ec2`.

Prima di installare un'immagine con Helm, visualizza il file `values.yaml` Helm per determinare la posizione dell'immagine. Ad esempio, il file values.yaml dell'helper di parametri del plugin CNI di Amazon VPC per Kubernetes include le seguenti righe.

```
image:
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. Estrai l'immagine del container specificata nel file manifesto.

   a. Se si esegue l'estrazione da un registro pubblico, come la Galleria pubblica di Amazon ECR, è possibile passare alla fase secondaria successiva poiché l'autenticazione non è richiesta. In questo esempio, esegui l'autenticazione a un registro privato di Amazon ECR contenente il repository per l'immagine dell'helper per i parametri CNI. Amazon EKS conserva l'immagine in ogni registro elencato nella pagina View Amazon container image registries for Amazon EKS add-ons. Puoi eseguire l'autenticazione a uno qualsiasi dei registri sostituendo *602401143452* e *region-code* con le informazioni di un registro differente. Esiste un registro separato per ciascuna Regione AWS in cui è supportato Amazon EKS.

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --
   password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
   ```

   b. Estrai l'immagine. In questo esempio, esegui l'estrazione dal registro a cui ti sei autenticato nel passaggio secondario precedente. Sostituisci *602401143452* e *region-code* con le informazioni che hai fornito nella fase secondaria precedente.

   ```
   docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
   helper:v1.12.6
   ```

4. Tagga l'immagine che hai estratto con il registro, il repository e il tag. L'esempio seguente presuppone che tu abbia estratto l'immagine dal file manifesto e la invierai al repository privato Amazon ECR creato nel primo passaggio. Sostituisci *111122223333* con l'ID del tuo account. Sostituisci *region-code* con la AWS regione in cui hai creato il tuo repository privato Amazon ECR.

   ```
   docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/
   cni-metrics-helper:v1.12.6
   ```

5. Esegui l'autenticazione al tuo registro. In questo esempio, esegui l'autenticazione al registro privato Amazon ECR creato nel primo passaggio. Per ulteriori informazioni, consulta la sezione Autenticazione del registro nella Guida per l'utente di Amazon Elastic Container Registry.

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. Invia l'immagine al tuo repository. In questo esempio, invii l'immagine al registro privato Amazon ECR creato nel primo passaggio. Per ulteriori informazioni, consulta la sezione Invio di un'immagine Docker nella Guida per l'utente di Amazon Elastic Container Registry.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

7. Aggiorna il file manifesto utilizzato per determinare l'immagine in un passaggio precedente con il `registry/repository:tag` per l'immagine che hai inviato. Se si esegue l'installazione con un grafico Helm, spesso è possibile specificare il `registry/repository:tag`. Quando installi il grafico, specifica il `registry/repository:tag` dell'immagine che hai inviato al repository.

# Visualizza i registri delle immagini del container per i componenti aggiuntivi di Amazon EKS

Quando implementi AWS Componenti aggiuntivi Amazon EKS nel tuo cluster, i nodi estraggono le immagini dei container necessarie dal registro specificato nel meccanismo di installazione del componente aggiuntivo, ad esempio un manifesto di installazione o un file `values.yaml` Helm. Le immagini vengono estratte da un repository privato Amazon ECR di Amazon EKS. Amazon EKS replica le immagini in un repository in ogni regione supportata AWS da Amazon EKS. I nodi possono estrarre l'immagine del container su Internet da uno dei seguenti registri. In alternativa, i tuoi nodi possono trasferire l'immagine sulla rete di Amazon se hai creato un endpoint VPC di interfaccia per Amazon ECR ()AWS PrivateLink nel tuo VPC. I registri richiedono l'autenticazione con un account IAM. AWS I tuoi nodi si autenticano utilizzando il ruolo IAM del nodo Amazon EKS, a cui sono associate le autorizzazioni nella policy IAM AmazonEC2ContainerRegistryReadOnlygestita.

| AWS Regione | Registro |
|---|---|
| af-south-1 | 877085696533.dkr.ecr.af-south-1.amazonaws.com |

| AWS Regione | Registro |
| --- | --- |
| ap-east-1 | 800184023465.dkr.ecr.ap-east-1.amazonaws.com |
| ap-east-2 | 533267051163.dkr. ecr.ap-east-2.amazonaws.com |
| ap-southeast-3 | 296578399912.dkr.ecr.ap-southeast-3.amazonaws.com |
| ap-south-2 | 900889452093.dkr.ecr.ap-south-2.amazonaws.com |
| ap-southeast-4 | 491585149902.dkr.ecr.ap-southeast-4.amazonaws.com |
| ap-southeast-5 | 151610086707.dkr. ecr.ap-southeast-5.amazonaws.com |
| ap-southeast-6 | 333609536671.dkr. ecr.ap-southeast-6.amazonaws.com |
| ap-south-1 | 602401143452.dkr.ecr.ap-south-1.amazonaws.com |
| ap-northeast-3 | 602401143452.dkr.ecr.ap-northeast-3.amazonaws.com |
| ap-northeast-2 | 602401143452.dkr.ecr.ap-northeast-2.amazonaws.com |
| ap-southeast-1 | 602401143452.dkr.ecr.ap-southeast-1.amazonaws.com |
| ap-southeast-2 | 602401143452.dkr.ecr.ap-southeast-2.amazonaws.com |
| ap-southeast-7 | 121268973566.dkr.ecr.ap-southeast-7.amazonaws.com |

| AWS Regione | Registro |
| --- | --- |
| ap-northeast-1 | 602401143452.dkr.ecr.ap-northeast-1.amazonaws.com |
| cn-north-1 | 918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn |
| cn-northwest-1 | 961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn |
| eu-central-1 | 602401143452.dkr.ecr.eu-central-1.amazonaws.com |
| eu-west-1 | 602401143452.dkr.ecr.eu-west-1.amazonaws.com |
| eu-west-2 | 602401143452.dkr.ecr.eu-west-2.amazonaws.com |
| eu-south-1 | 590381155156.dkr.ecr.eu-south-1.amazonaws.com |
| eu-west-3 | 602401143452.dkr.ecr.eu-west-3.amazonaws.com |
| eu-south-2 | 455263428931.dkr.ecr.eu-south-2.amazonaws.com |
| eu-north-1 | 602401143452.dkr.ecr.eu-north-1.amazonaws.com |
| eu-central-2 | 900612956339.dkr.ecr.eu-central-2.amazonaws.com |
| il-central-1 | 066635153087.dkr.ecr.il-central-1.amazonaws.com |
| mx-central-1 | 730335286997.dkr.ecr.mx-central-1.amazonaws.com |

| AWS Regione | Registro |
| --- | --- |
| me-south-1 | 558608220178.dkr.ecr.me-south-1.amazonaws.com |
| me-central-1 | 759879836304.dkr.ecr.me-central-1.amazonaws.com |
| us-east-1 | 602401143452.dkr.ecr.us-east-1.amazonaws.com |
| us-east-2 | 602401143452.dkr.ecr.us-east-2.amazonaws.com |
| us-west-1 | 602401143452.dkr.ecr.us-west-1.amazonaws.com |
| us-west-2 | 602401143452.dkr.ecr.us-west-2.amazonaws.com |
| ca-central-1 | 602401143452.dkr.ecr.ca-central-1.amazonaws.com |
| ca-west-1 | 761377655185.dkr.ecr.ca-west-1.amazonaws.com |
| sa-east-1 | 602401143452.dkr.ecr.sa-east-1.amazonaws.com |
| us-gov-east-1 | 151742754352.dkr.ecr.us-gov-east-1.amazonaws.com |
| us-gov-west-1 | 013241004608.dkr.ecr.us-gov-west-1.amazonaws.com |
| eusc-de-east-1 | 877088126301.dkr.ecr.eusc-de-east-1.amazonaws.eu |

# Componenti aggiuntivi Amazon EKS

Un componente aggiuntivo è un software che fornisce funzionalità operative di supporto alle applicazioni Kubernetes, ma non è specifico per l'applicazione. Ciò include software come agenti di osservabilità o driver Kubernetes che consentono al cluster di interagire con le risorse sottostanti AWS per il networking, l'elaborazione e lo storage. Add-on il software è in genere creato e gestito dalla community Kubernetes, da fornitori di cloud o da fornitori di terze parti. AWS Per ogni cluster, Amazon EKS installa automaticamente componenti aggiuntivi autogestiti come il plugin CNI di Amazon VPC per Kubernetes, `kube-proxy`, e CoreDNS. Tenere presente che il componente aggiuntivo CNI VPC non è compatibile con Amazon EKS Hybrid Nodes e non viene implementato su nodi ibridi. È possibile modificare la configurazione predefinita dei componenti aggiuntivi e aggiornarli quando lo desideri.

I componenti aggiuntivi di Amazon EKS forniscono l'installazione e la gestione di un set curato di componenti aggiuntivi per i cluster Amazon EKS. Tutti i componenti aggiuntivi di Amazon EKS includono le ultime patch di sicurezza e correzioni di bug e sono convalidati per funzionare con Amazon EKS. AWS I componenti aggiuntivi Amazon EKS permettono di garantire costantemente sicurezza e stabilità ai cluster Amazon EKS e di ridurre la quantità di lavoro necessaria per installare, configurare e aggiornare i componenti aggiuntivi. Se un componente aggiuntivo autogestito, ad esempio `kube-proxy` è già in esecuzione nel cluster ed è disponibile come componente aggiuntivo di Amazon EKS, è possibile installare il componente aggiuntivo `kube-proxy` Amazon EKS per iniziare a beneficiare delle funzionalità dei componenti aggiuntivi di Amazon EKS.

Tramite l'API Amazon EKS. è possibile aggiornare specifici campi di configurazione gestiti da Amazon EKS per i componenti aggiuntivi Amazon EKS. É anche possibile modificare i campi di configurazione non gestiti da Amazon EKS direttamente all'interno del cluster Kubernetes una volta avviato il componente aggiuntivo. Ciò include, ove applicabile, la definizione di campi di configurazione specifici per un componente aggiuntivo. Una volta apportate, queste modifiche non vengono sovrascritte da Amazon EKS. Ciò è reso possibile utilizzando la caratteristica applicazione lato server Kubernetes. Per ulteriori informazioni, consulta the section called “Campi che puoi personalizzare”.

È possibile utilizzare i componenti aggiuntivi di Amazon EKS con qualsiasi tipo di nodo Amazon EKS. Per ulteriori informazioni, consulta Gestione del calcolo.

Puoi aggiungere, aggiornare o eliminare i componenti aggiuntivi di Amazon EKS utilizzando l'API Amazon EKS, Console di gestione AWS l'interfaccia a riga di comando e. AWS `eksctl` La creazione di componenti aggiuntivi di Amazon EKS è possibile anche utilizzando AWS CloudFormation.

# Considerazioni

Tenere in considerazione quanto segue quando si utilizzano i componenti aggiuntivi Amazon EKS:

- Per configurare i componenti aggiuntivi per il cluster, il principale IAM deve disporre delle autorizzazioni IAM per l'utilizzo dei componenti aggiuntivi. Per ulteriori informazioni, consulta le operazioni con Addon nel loro nome in Operazioni definite da Amazon Elastic Kubernetes Service.
- I componenti aggiuntivi di Amazon EKS vengono eseguiti sui nodi con provisioning o configurazione per il cluster. I tipi di nodi includono istanze Amazon EC2, Fargate e nodi ibridi.
- È possibile modificare i campi non gestiti da Amazon EKS per personalizzare l'installazione di un componente aggiuntivo Amazon EKS. Per ulteriori informazioni, consulta the section called "Campi che puoi personalizzare".
- Se crei un cluster con Console di gestione AWS, aggiunge Console di gestione AWS automaticamente i componenti aggiuntivi Amazon EKS`kube-proxy`, Amazon VPC CNI per Kubernetes e CoreDNS al tuo cluster. Se crei il cluster utilizzando `eksctl` senza un `config` file, `eksctl` installa il plug-in Amazon VPC CNI per Kubernetes`kube-proxy`, CoreDNS e metrics-server come componenti aggiuntivi di Amazon EKS a partire dalla versione 0.184.0. Per le versioni precedenti o se utilizzi un altro strumento`eksctl`, installerai invece il plug-in Amazon VPC CNI per Kubernetes e i componenti aggiuntivi CoreDNS a gestione autonoma`kube-proxy`. Puoi gestirli in autonomia o aggiungere manualmente i componenti aggiuntivi di Amazon EKS dopo la creazione del cluster. Indipendentemente dal metodo utilizzato per creare il cluster, il plug-in Amazon VPC CNI per Kubernetes non si installa su nodi ibridi.
- `eks:addon-cluster-admin` `ClusterRoleBinding` associa `cluster-admin` `ClusterRole` all'identità Kubernetes `eks:addon-manager`. Il ruolo dispone delle autorizzazioni necessarie all'identità `eks:addon-manager` per creare namespace Kubernetes e per installare componenti aggiuntivi nei namespace. Se il `ClusterRoleBinding` `eks:addon-cluster-admin` viene rimosso, il cluster Amazon EKS continua a funzionare, ma Amazon EKS non è più in grado di gestire i componenti aggiuntivi. Tutti i cluster a partire dalle seguenti versioni della piattaforma utilizzano il nuovo `ClusterRoleBinding`.
- Un sottoinsieme di componenti aggiuntivi EKS di è stato convalidato per la compatibilità con Amazon EKS AWS Hybrid Nodes. Per ulteriori informazioni, consultare la tabella di compatibilità alla pagina the section called " AWS componenti aggiuntivi".

# Namespace personalizzato per componenti aggiuntivi

Per la community e AWS i componenti aggiuntivi, puoi facoltativamente specificare uno spazio dei nomi personalizzato durante la creazione del componente aggiuntivo. Una volta installato un componente aggiuntivo in un namespace specifico, è necessario rimuovere e creare nuovamente il componente per modificarne il namespace.

Se non viene specificato un namespace, verrà utilizzato il namespace predefinito del componente aggiuntivo.

Utilizzare namespace personalizzati per una migliore organizzazione e un miglior isolamento degli oggetti del componente aggiuntivo all'interno del cluster EKS. Questa flessibilità consente di conformare i componenti aggiuntivi alle esigenze operative e alla strategia di namespace esistente.

È possibile impostare un namespace personalizzato durante la creazione di un componente aggiuntivo. Per ulteriori informazioni, consulta the section called "Creare un componente aggiuntivo".

## Ottenere un namespace predefinito di un componente aggiuntivo

Il namespace predefinito di un componente aggiuntivo è il namespace in cui sarà installato se non ne viene specificato uno.

Utilizzare il seguente comando per ottenere il namespace predefinito di un componente aggiuntivo:

```
aws eks describe-addon-versions --addon-name <addon-name> --query
 "addons[].defaultNamespace"
```

Output di esempio:

```
[
    "kube-system"
]
```

# Considerazioni per la modalità automatica di Amazon EKS

La modalità automatica di Amazon EKS include funzionalità che garantiscono l'operatività essenziale del cluster, tra cui:

- Reti pod
- Reti di servizio

- DNS del cluster
- Dimensionamento automatico
- Storage a blocchi
- Controller del bilanciatore del carico
- Agente Pod Identity
- Agente di monitoraggio del nodo

Con il calcolo in modalità automatica, molti dei componenti aggiuntivi EKS più utilizzati diventano ridondanti, come:

- CNI di Amazon VPC
- kube-proxy
- CoreDNS
- Driver CSI per Amazon EBS
- EKS Pod Identity Agent

Tuttavia, se il cluster combina la modalità Auto con altre opzioni di calcolo come istanze EC2 autogestite, Managed Node Groups o Fargate, questi componenti aggiuntivi rimangono necessari. AWS AWS dispone di componenti aggiuntivi EKS migliorati con regole anti-affinità che assicurano automaticamente che i pod aggiuntivi siano pianificati solo sui tipi di calcolo supportati. Inoltre, ora gli utenti possono ora utilizzare l'API `DescribeAddonVersions` dei componenti aggiuntivi EKS per verificare i tipi di calcolo supportati (computeTypes) per ciascun componente aggiuntivo e le relative versioni specifiche. Inoltre, con la modalità EKS Auto, i controller sopra elencati funzionano su un'infrastruttura di proprietà. AWS Quindi, potresti non vederli nemmeno nei tuoi account a meno che tu non stia utilizzando la modalità automatica EKS con altri tipi di calcolo, nel qual caso vedrai i controller che hai installato sul tuo cluster.

Se si intende abilitare la modalità automatica EKS su un cluster esistente, potrebbe essere necessario aggiornare la versione di alcuni componenti aggiuntivi. Per ulteriori informazioni, consultare the section called "Versioni richieste del componente aggiuntivo" per la modalità automatica EKS.

## Supporto

AWS pubblica diversi tipi di componenti aggiuntivi con diversi livelli di supporto.

- AWS Add-ons:  Questi componenti aggiuntivi sono creati e completamente supportati da. AWS
  - Usa un AWS componente aggiuntivo per lavorare con altri AWS servizi, come Amazon EFS.
  - Per ulteriori informazioni, consulta the section called " AWS componenti aggiuntivi".
- AWS Marketplace Add-ons:  questi componenti aggiuntivi vengono scansionati AWS e supportati da un partner indipendente. AWS
  - Utilizzare un componente aggiuntivo del marketplace per aggiungere funzionalità preziose e sofisticate al cluster, come il monitoraggio con Splunk.
  - Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi di Marketplace".
- Comunità Add-ons: questi componenti aggiuntivi sono controllati AWS ma supportati dalla community open source.
  - Utilizzare un componente aggiuntivo della community per ridurre la complessità dell'installazione di software open source comuni, come Kubernetes Metrics Server.
  - I componenti aggiuntivi della community sono confezionati partendo dai sorgenti di. AWS AWS convalida i componenti aggiuntivi della community solo per la compatibilità delle versioni.
  - Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi della community".

La tabella seguente illustra la portata del supporto per ogni tipo di componente aggiuntivo:

| Categoria | Funzionalità | AWS componenti aggiuntivi | AWS componenti aggiuntivi del Marketplace | Componenti aggiuntivi della community |
|---|---|---|---|---|
| Sviluppo | Costruito da AWS | Sì | No | Sì |
| Sviluppo | Convalidato da AWS | Sì | No | Sì* |
| Sviluppo | Convalidato dal partner AWS | No | Sì | No |
| Maintenance (Manutenzione) | Scansionato da AWS | Sì | Sì | Sì |

| Categoria | Funzionalità | AWS componenti aggiuntivi | AWS componenti aggiuntivi del Marketplace | Componenti aggiuntivi della community |
|---|---|---|---|---|
| Maintenance (Manutenzione) | Patchato da AWS | Sì | No | Sì |
| Maintenance (Manutenzione) | Patchato dal partner AWS | No | Sì | No |
| Distribuzione | Pubblicato da AWS | Sì | No | Sì |
| Distribuzione | Pubblicato da AWS Partner | No | Sì | No |
| Supporto | Supporto di installazione di base a cura di AWS | Sì | Sì | Sì |
| Supporto | AWS Supporto completo | Sì | No | No |
| Supporto | Supporto completo AWS per i partner | No | Sì | No |

*: la convalida dei componenti aggiuntivi della community include solo la compatibilità delle versioni di Kubernetes. Ad esempio, se installi un componente aggiuntivo della community su un cluster, AWS verifica se è compatibile con la versione Kubernetes del cluster.

AWS I componenti aggiuntivi di Marketplace possono scaricare dipendenze software aggiuntive da fonti esterne esterne a. AWS Queste dipendenze esterne non vengono scansionate o convalidate da. AWS Considera i tuoi requisiti di sicurezza quando distribuisci componenti aggiuntivi di AWS Marketplace che recuperano le dipendenze esterne.

# AWS componenti aggiuntivi

Nel tuo cluster puoi creare i seguenti componenti aggiuntivi di Amazon EKS. Puoi visualizzare l'elenco più aggiornato dei componenti aggiuntivi disponibili utilizzando`eksctl`, il Console di gestione AWS, o la CLI AWS . Per visualizzare tutti i componenti aggiuntivi disponibili o per installare un componente aggiuntivo, consulta la pagina the section called "Creare un componente aggiuntivo". Se un componente aggiuntivo richiede autorizzazioni IAM, è necessario disporre di un provider OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". È possibile creare o eliminare un componente aggiuntivo dopo averlo installato. Per ulteriori informazioni, consulta the section called "Aggiornamento di un componente aggiuntivo" o the section called "Rimozione di un componente aggiuntivo". Per ulteriori informazioni su considerazioni specifiche in merito all'esecuzione di componenti aggiuntivi EKS con Amazon EKS Hybrid Nodes, consulta the section called "Configurare i componenti aggiuntivi".

È possibile utilizzare uno dei seguenti componenti aggiuntivi di Amazon EKS.

| Description | Ulteriori informazioni | Tipi di elaborazione compatibili |
|---|---|---|
| Fornire una rete VPC nativa per il tuo cluster | the section called "Plug-in della CNI di Amazon VPC per Kubernetes" | EC2 |
| Un server DNS flessibile ed estensibile che può fungere da cluster DNS Kubernetes | the section called "CoreDNS" | EC2, Fargate, modalità automatica EKS, nodi ibridi EKS |
| Mantenere le regole di rete su ciascun nodo Amazon EC2 | the section called "`Kube-proxy` " | Nodi ibridi EC2, EKS |
| Fornisci l'archiviazione Amazon EBS per il cluster | the section called "Driver CSI per Amazon EBS" | EC2 |
| Fornisci lo storage di istanze Amazon EC2 per il tuo cluster | the section called "Driver CSI di Amazon EC2 Instance Store" | EC2 |

| Description | Ulteriori informazioni | Tipi di elaborazione compatibili |
| --- | --- | --- |
| Fornire l'archiviazione Amazon EBS per il cluster | the section called "Driver CSI per Amazon EFS" | EC2, modalità automatica EKS |
| Fornisci lo storage di Amazon S3 Files per il tuo cluster | the section called "Driver CSI per Amazon EFS" | EC2, modalità automatica EKS |
| Fornire l'archiviazione di Amazon FSx per Lustre per il cluster | the section called "Driver CSI per Amazon FSx" | EC2, modalità automatica EKS |
| Fornire l'archiviazione Amazon S3 per il cluster | the section called "Driver CSI di Mountpoint per Amazon S3" | EC2, modalità automatica EKS |
| Rilevare ulteriori problemi di integrità dei nodi | the section called "Agente di monitoraggio del nodo" | Nodi ibridi EC2, EKS |
| Abilitare l'utilizzo della funzionalità snapshot nei driver CSI compatibili, come il driver CSI per Amazon EBS. | the section called "Controller di snapshot CSI" | EC2, Fargate, modalità automatica EKS, nodi ibridi EKS |
| SageMaker HyperPod la governance delle attività ottimizza l'allocazione e l'utilizzo delle risorse di calcolo tra i team nei cluster Amazon EKS, risolvendo le inefficienze nella definizione delle priorità delle attività e nella condivisione delle risorse. | the section called "Governance delle SageMaker HyperPod attività di Amazon" | EC2, modalità automatica EKS |
| Amazon SageMaker HyperPod Observability fornisce funzionalità complete di monitoraggio e osservabilità AddOn per i cluster. HyperPod | the section called "Amazon Observability SageMaker HyperPod Add-on" | EC2, modalità automatica EKS |

| Description | Ulteriori informazioni | Tipi di elaborazione compatibili |
| --- | --- | --- |
| L'operatore SageMaker HyperPod di formazione Amazon consente una formazione distribuita efficiente sui cluster Amazon EKS con funzionalità avanzate di pianificazione e gestione delle risorse. | the section called "Operatore di formazione Amazon SageMaker HyperPod" | EC2, modalità automatica EKS |
| Amazon SageMaker HyperPod Inference Operator consente l'implementazione e la gestione di carichi di lavoro di inferenza AI ad alte prestazioni con utilizzo ottimizzato delle risorse ed efficienza dei costi. | the section called "Operatore di SageMaker HyperPod inferenza Amazon" | EC2, modalità automatica EKS |
| Un agente Kubernetes che raccoglie e riporta i dati del flusso di rete ad Amazon CloudWatch, consentendo un monitoraggio completo delle connessioni TCP tra i nodi del cluster. | the section called "AWS Network Flow Monitor Agent" | EC2, modalità automatica EKS |
| Distribuzione sicura, pronta per la produzione e supportata del progetto AWS OpenTelemetry | the section called "AWS Distro per OpenTelemetry" | EC2, Fargate, modalità automatica EKS, nodi ibridi EKS |

| Description | Ulteriori informazioni | Tipi di elaborazione compatibili |
| --- | --- | --- |
| Servizio di monitoraggio della sicurezza che analizza ed elabora le fonti di dati fondamentali, inclusi gli eventi di AWS CloudTrail gestione e i log di flusso di Amazon VPC. Amazon elabora GuardDuty anche funzionalità, come i log di controllo di Kubernetes e il monitoraggio del runtime | the section called "Agente Amazon GuardDuty" | EC2, modalità automatica EKS |
| Servizio di monitoraggio e osservabilità fornito da AWS. Questo componente aggiuntivo installa l' CloudWatch agente e abilita sia CloudWatch Application Signals che CloudWatch Container Insights con una migliore osservabilità per Amazon EKS | the section called "Agente Amazon Observability CloudWatch" | EC2, modalità automatica EKS, nodi ibridi EKS |
| Possibilità di gestire le credenziali per le applicazioni, analogamente al modo in cui i profili dell'istanza EC2 forniscono le credenziali alle istanze EC2 | the section called "EKS Pod Identity Agent" | Nodi ibridi EC2, EKS |
| Abilita cert-manager per emettere certificati da Private CA X.509 . AWS Richiede cert-manager. | the section called "AWS Connettore CA privato per Kubernetes" | EC2, Fargate, modalità automatica EKS, nodi ibridi EKS |

| Description | Ulteriori informazioni | Tipi di elaborazione compatibili |
| --- | --- | --- |
| Genera metriche Prometheus sulle prestazioni dei dispositivi di rete SR-IOV | the section called "SR-IOV Network Metrics Exporter" | EC2 |
| Recupera i AWS segreti da Secrets Manager e i parametri da AWS Systems Manager Parameter Store e montali come file nei pod Kubernetes. | the section called "AWS Fornitore di driver CSI Secrets Store" | EC2, modalità automatica EKS, nodi ibridi EKS |
| Con Spaces, puoi creare JupyterLab e gestire applicazioni Code Editor per eseguire carichi di lavoro ML interattivi. | the section called "Amazon Spaces SageMaker" | Hyperpod |

## Plug-in della CNI di Amazon VPC per Kubernetes

Il componente aggiuntivo Amazon EKS del plugin CNI di Amazon VPC per Kubernetes è un plugin di interfaccia di rete container (CNI) Kubernetes che fornisce una rete VPC nativa per il cluster. Il tipo autogestito o gestito di questo componente aggiuntivo è installato su ogni nodo Amazon EC2 per impostazione predefinita. Per ulteriori informazioni, consulta Plugin di interfaccia di rete container (CNI) Kubernetes.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `vpc-cni`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

Se il cluster utilizza la famiglia `IPv4`, sono necessarie le autorizzazioni in AmazonEKS_CNI_Policy. Se il cluster utilizza la famiglia `IPv6`, è necessario creare una policy IAM con le autorizzazioni in modalità IPv6. È possibile creare un ruolo IAM, collegare una delle policy e annotare l'account del servizio Kubernetes utilizzato dal componente aggiuntivo con il comando seguente.

Sostituisci *`my-cluster`* con il nome del cluster e *`AmazonEKSVPCCNIRole`* con il nome del ruolo. Se il tuo cluster utilizza la famiglia `IPv6`, sostituisci *`AmazonEKS_CNI_Policy`* con il nome della policy che hai creato. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo, collegarvi la policy e annotare l'account del servizio Kubernetes, consulta the section called "Assegnazione di un ruolo IAM".

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

Aggiornare le informazioni

Puoi aggiornare solo una versione secondaria alla volta. Ad esempio, se la versione corrente è `1.28.`*`x`*`-eksbuild.`*`y`* e desideri aggiornarla a `1.30.`*`x`*`-eksbuild.`*`y`* , devi prima eseguire l'aggiornamento a `1.29.`*`x`*`-eksbuild.`*`y`* e quindi l'aggiornamento a `1.30.`*`x`*`-eksbuild.`*`y`* . Per ulteriori informazioni sull'aggiornamento del componente aggiuntivo, consulta the section called "Aggiornamento (componente aggiuntivo EKS)".

## CoreDNS

Il componente aggiuntivo CoreDNS per Amazon EKS è un server DNS flessibile ed estensibile che può fungere da DNS del cluster di Kubernetes. Il tipo autogestito o gestito di questo componente aggiuntivo è stato installato, per impostazione predefinita, al momento della creazione del cluster. Quando si avvia un cluster Amazon EKS con almeno un nodo, due repliche dell'immagine CoreDNS vengono distribuite per impostazione predefinita, indipendentemente dal numero di nodi distribuiti nel cluster. I contenitori CoreDNS forniscono la risoluzione dei nomi per tutti i pod del cluster. È possibile implementare i pod CoreDNS ai nodi Fargate se il cluster include un profilo Fargate con uno namespace che corrisponde al namespace per l'implementazione di CoreDNS. Per ulteriori informazioni, consulta the section called "Definisci i profili".

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `coredns`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per ulteriori informazioni su CoreDNS, consulta Utilizzo di CoreDNS per il rilevamento servizi e Personalizzazione del servizio DNS nella documentazione di Kubernetes.

### `Kube-proxy`

Il componente aggiuntivo di Amazon EKS `Kube-proxy` mantiene le regole di rete su ciascun nodo Amazon EC2. Consente la comunicazione di rete con i tuoi pod. Il tipo autogestito o gestito di questo componente aggiuntivo è installato su ogni nodo Amazon EC2 nel cluster per impostazione predefinita.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `kube-proxy`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Aggiornare le informazioni

Prima di aggiornare la versione corrente, considera i seguenti requisiti:

- `Kube-proxy` su un cluster Amazon EKS ha la stessa policy di compatibilità e disallineamento di Kubernetes.

Informazioni aggiuntive

Per ulteriori informazioni su `kube-proxy`, consulta kube-proxy nella documentazione Kubernetes.

## Driver CSI per Amazon EBS

Il componente aggiuntivo di Amazon EKS per il driver CSI per Amazon EBS è un plugin di interfaccia di archiviazione del container (CSI) Kubernetes che fornisce archiviazione Amazon EBS per il cluster.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. La modalità automatica include una funzionalità di archiviazione a blocchi. Per ulteriori informazioni, consulta the section called "Implementazione di un carico di lavoro stateful".

Il nome del componente aggiuntivo di Amazon EKS è `aws-ebs-csi-driver`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo usa i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA". Questo componente aggiuntivo richiede una delle seguenti politiche AWS gestite: AmazonEBSCSIDriverPolicyV2 per l'ambito basato su tag, AmazonEBSCSIDriverEKSClusterScopedPolicy per l'isolamento con ambito di cluster o AmazonEBSCSIDriverPolicy se non desideri alcuna restrizione basata sui tag. Puoi creare un ruolo IAM e collegarvi la policy gestita con il seguente comando. Sostituisci *`my-cluster`* con il nome del cluster e *`AmazonEKS_EBS_CSI_DriverRole`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se hai la necessità di utilizzare uno strumento diverso o una chiave KMS personalizzata per la crittografia, consulta la sezione the section called "Passaggio 1: Creazione di un ruolo IAM".

In caso di migrazione da, consulta la sezione EBS CSI Driver policy `AmazonEBSCSIDriverPolicy` migration. https://github.com/kubernetes-sigs/aws-ebs-csi-driver/issues/2918

```
eksctl create iamserviceaccount \
```

```
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicyV2 \
    --approve
```

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Amazon EBS".

## Driver CSI di Amazon EC2 Instance Store

Il driver Amazon EKS per il driver Amazon EC2 Instance Store CSI è un plug-in Kubernetes Container Storage Interface (CSI) che fornisce lo storage NVMe dell'Instance Store di Amazon EC2 per il tuo cluster.

Il nome del componente aggiuntivo di Amazon EKS è `aws-ec2-local-instance-store-csi-driver`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione. Il driver interagisce con i dispositivi NVMe di backend tramite operazioni Linux locali e non effettua chiamate API. AWS

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Archivio dell'istanza Amazon EC2".

## Driver CSI per Amazon EFS

Il driver Amazon EFS CSI (componente aggiuntivo Amazon EKS) è un plug-in Kubernetes Container Storage Interface (CSI) che fornisce lo storage di file Amazon EFS e Amazon S3 per il tuo cluster.

Il nome del componente aggiuntivo di Amazon EKS è `aws-efs-csi-driver`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo usa i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

La policy AWS gestita specifica di cui hai bisogno dipende dal tipo di file system che desideri utilizzare:

- Solo per i file system Amazon EFS: AmazonEFSCSIDriverPolicy allega la policy gestita.
- Solo per i file system Amazon S3: allega la policy `AmazonS3FilesCSIDriverPolicy` gestita.
- Sia per i file system Amazon EFS che Amazon S3: allega sia le policy gestite che quelle `AmazonEFSCSIDriverPolicy` `AmazonS3FilesCSIDriverPolicy` gestite.

Puoi creare un ruolo IAM e collegarvi la policy gestita con i seguenti comandi. Sostituisci *my-cluster* con il nome del cluster e *AmazonEKS_EFS_CSI_DriverRole* con il nome del ruolo. L'esempio seguente allega i file system `AmazonEFSCSIDriverPolicy` per Amazon EFS. Se utilizzi un file system Amazon S3, sostituisci la policy ARN con. `arn:aws: iam::aws:policy/service-role/AmazonS3FilesCSIDriverPolicy` Se utilizzi entrambi i tipi di file system, aggiungi un `--attach-policy-arn` flag aggiuntivo con la seconda policy ARN. Questi comandi richiedono l'installazione di eksctl sul tuo dispositivo. Se devi usare uno strumento diverso, consulta the section called "Passaggio 1: Creazione di un ruolo IAM" per Amazon EFS o the section called "Fase 1: Creare ruoli IAM" per Amazon S3 Files.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

> **Note**
>
> L'esempio precedente configura `efs-csi-controller-sa` solo. Se utilizzi i file system Amazon S3, devi anche configurare. `efs-csi-node-sa` Consulta the section called "Fase 1: Creare ruoli IAM" la configurazione completa di S3 Files IAM.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Amazon EFS".

## Driver CSI per Amazon FSx

Il componente aggiuntivo di Amazon EKS per il driver CSI per Amazon FSx è un plugin di interfaccia di archiviazione del container (CSI) Kubernetes che fornisce archiviazione Amazon FSx per Lustre per il cluster.

Il nome del componente aggiuntivo di Amazon EKS è `aws-fsx-csi-driver`.

> **Note**
>
> - Pre-existing Le installazioni dei driver Amazon FSx CSI nel cluster possono causare errori di installazione dei componenti aggiuntivi. Quando tenti di installare la versione del componente aggiuntivo Amazon EKS mentre esiste un driver CSI FSx non EKS, l'installazione avrà esito negativo a causa di conflitti di risorse. Usa il flag `OVERWRITE` durante l'installazione per risolvere il problema:
>
>   ```
>   aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
>    --resolve-conflicts OVERWRITE
>   ```
>
> - Il componente aggiuntivo Amazon FSx CSI Driver EKS supporta l'autenticazione tramite EKS Pod Identity o IAM Roles for Service Accounts (IRSA). Per utilizzare EKS Pod Identity, installa l'agente Pod Identity prima o dopo l'implementazione del componente aggiuntivo FSx CSI Driver. Per ulteriori informazioni, consulta the section called "Configurazione dell'agente". Per utilizzare invece IRSA, vedere. the section called "Gestore di identità OIDC IAM"

Autorizzazioni IAM richieste

Questo componente aggiuntivo usa i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA". Le autorizzazioni nella policy AmazonFSxFullAccess AWS gestita sono obbligatorie. Puoi creare un ruolo IAM e collegarvi la policy gestita con il seguente comando. Sostituisci *my-cluster* con il nome del cluster e *AmazonEKS_FSx_CSI_DriverRole* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo.

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Amazon FSx per Lustre".

## Driver CSI di Mountpoint per Amazon S3

Il componente aggiuntivo Mountpoint per Amazon S3 di Amazon EKS per il driver CSI è un plugin di interfaccia di archiviazione del container (CSI) Kubernetes che fornisce archiviazione Amazon S3 per il cluster.

Il nome del componente aggiuntivo di Amazon EKS è `aws-mountpoint-s3-csi-driver`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

Il ruolo IAM creato richiederà una policy che fornisca l'accesso a S3. Segui i Suggerimenti sulle autorizzazioni IAM di Mountpoint durante la creazione della policy. In alternativa, è possibile utilizzare la policy AWS gestita AmazonS3FullAccess, ma questa policy gestita concede più autorizzazioni di quelle necessarie per Mountpoint.

Puoi creare un ruolo IAM e collegarvi la policy gestita con i seguenti comandi. Sostituiscila *my-cluster* con il nome del tuo cluster, *region-code* con il codice AWS regionale corretto, *AmazonEKS_S3_CSI_DriverRole* con il nome del tuo ruolo e *AmazonEKS_S3_CSI_DriverRole_ARN* con il ruolo ARN. Questi comandi richiedono l'installazione di eksctl sul tuo dispositivo. Per istruzioni sull'uso della console IAM o della AWS CLI, consulta. the section called "Fase 2: creazione di un ruolo IAM"

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Mountpoint per Amazon S3".

## Controller di snapshot CSI

Il controller di snapshot dell'interfaccia di archiviazione di container (CSI) consente l'uso della funzionalità snapshot nei driver CSI compatibili, come il driver CSI di Amazon EBS.

Il nome del componente aggiuntivo di Amazon EKS è `snapshot-controller`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Controller snapshot CSI".

## Governance delle SageMaker HyperPod attività di Amazon

SageMaker HyperPod la governance delle attività è un solido sistema di gestione progettato per semplificare l'allocazione delle risorse e garantire un utilizzo efficiente delle risorse di calcolo tra team e progetti per i tuoi cluster Amazon EKS. Questa funzionalità offre agli amministratori la possibilità di impostare:

- Livelli di priorità per varie attività
- Allocazione delle risorse di calcolo per ogni team
- In che modo ogni team presta e prende in prestito risorse di calcolo inattive
- Se un team rende prerilasciabili le proprie attività

HyperPod la governance delle attività fornisce anche Amazon EKS Cluster Observability, che offre visibilità in tempo reale sulla capacità del cluster. Ciò include la disponibilità e l'utilizzo delle risorse di calcolo, l'allocazione e l'utilizzo del team e le informazioni sull'esecuzione delle attività e sui tempi di attesa, per consentirti di prendere decisioni informate e gestire in modo proattivo le risorse.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-taskgovernance`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per saperne di più sull'add-on, consulta SageMaker HyperPod la sezione Task Governance

## Amazon Observability SageMaker HyperPod Add-on

Amazon SageMaker HyperPod Observability Add-on fornisce funzionalità complete di monitoraggio e osservabilità per i cluster. HyperPod Questo componente aggiuntivo implementa e gestisce automaticamente i componenti di monitoraggio essenziali, tra cui node exporter, DCGM exporter, kube-state-metrics ed EFA exporter. Raccoglie e inoltra i parametri a un'istanza Prometheus gestita da Amazon (AMP) designata dal cliente ed espone un endpoint OTLP per metriche personalizzate e acquisizione di eventi dai job di addestramento dei clienti.

L'add-on si integra con l' HyperPod ecosistema più ampio estrapolando le metriche da vari componenti, tra cui il componente aggiuntivo HyperPod Task Governance, Training Operator,

Kubeflow e KEDA. HyperPod Tutte le metriche raccolte sono centralizzate in Prometheus gestito da Amazon, che consente ai clienti di ottenere una visione unificata dell'osservabilità tramite i dashboard di Grafana gestito da Amazon. Ciò fornisce una visibilità end-to-end sullo stato del cluster, sull'utilizzo delle risorse e sulle prestazioni dei lavori di formazione nell'intero ambiente. HyperPod

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-observability`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA". Di seguito sono elencate le policy gestite richieste:

- `AmazonPrometheusRemoteWriteAccess`: per la scrittura remota delle metriche a partire dal cluster fino ad AMP
- `CloudWatchAgentServerPolicy`- per la scrittura remota dei log dal cluster a CloudWatch

Informazioni aggiuntive

Per saperne di più sull'add-on e sulle sue funzionalità, vedi SageMaker HyperPod Observability.

## Operatore di formazione Amazon SageMaker HyperPod

L'operatore di SageMaker HyperPod formazione Amazon ti aiuta ad accelerare lo sviluppo di modelli di intelligenza artificiale generativi gestendo in modo efficiente la formazione distribuita su cluster di GPU di grandi dimensioni. Introduce funzionalità intelligenti di ripristino dai guasti, rilevamento delle interruzioni e gestione a livello di processo che riducono al minimo le interruzioni dell'addestramento e riducono i costi. A differenza dell'infrastruttura di addestramento tradizionale che richiede il riavvio completo del processo in caso di guasto, questo operatore implementa un ripristino chirurgico del processo per garantire il corretto funzionamento dei job di addestramento.

L'operatore collabora anche con le funzioni HyperPod di monitoraggio e osservabilità dello stato di salute, fornendo visibilità in tempo reale sull'esecuzione della formazione e il monitoraggio automatico di metriche critiche come i picchi di perdita e il degrado della produttività. Puoi definire le policy di ripristino tramite semplici configurazioni YAML senza modifiche al codice, che consentono di rispondere rapidamente e ripristinare gli stati irreversibili dell'addestramento. Queste funzionalità di monitoraggio e ripristino interagiscono per garantire prestazioni di addestramento ottimali riducendo al minimo il sovraccarico operativo.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-training-operator`.

Per ulteriori informazioni, consulta Using the HyperPod training operator nella Amazon Developer Guide. SageMaker

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede le autorizzazioni IAM e utilizza EKS Pod Identity.

AWS suggerisce la policy `AmazonSageMakerHyperPodTrainingOperatorAccess` gestita.

Per ulteriori informazioni, consulta Installing the training operator nella Amazon SageMaker Developer Guide.

Informazioni aggiuntive

Per saperne di più sul componente aggiuntivo, consulta SageMaker HyperPod Training operator.

## Operatore di SageMaker HyperPod inferenza Amazon

Amazon SageMaker HyperPod offre un'esperienza end-to-end che supporta l'intero ciclo di vita dello sviluppo dell'IA, dalla sperimentazione e formazione interattive ai flussi di lavoro di inferenza e post-formazione. Ora fornisce una piattaforma di inferenza completa che combina la flessibilità di Kubernetes con l'eccellenza operativa di un'esperienza gestita. Implementa, scala e ottimizza i tuoi modelli GenAI con un'affidabilità di livello aziendale utilizzando lo stesso calcolo per l'intero ciclo di vita del modello. HyperPod

Amazon SageMaker HyperPod offre interfacce di distribuzione flessibili che consentono di distribuire modelli tramite diversi metodi, tra cui kubectl, Python SDK, Amazon Studio UI o CLI. SageMaker HyperPod La funzionalità offre funzionalità avanzate di scalabilità automatica con allocazione dinamica delle risorse che si adatta automaticamente in base alla domanda. Inoltre, include funzionalità complete di osservabilità e monitoraggio che tengono traccia di metriche critiche come il tempo al primo token (Time To First Token), la latenza e l'utilizzo della GPU per aiutarti a ottimizzare le prestazioni.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-inference`.

Metodi di installazione

È possibile installare questo componente aggiuntivo utilizzando uno dei seguenti metodi:

- SageMaker Console (consigliata): fornisce un'esperienza di installazione semplificata con configurazione guidata.
- Add-ons Console o CLI EKS: richiede l'installazione manuale dei componenti aggiuntivi di dipendenza prima di installare l'operatore di inferenza. Consulta la sezione sui prerequisiti di seguito per le dipendenze richieste.

Prerequisiti

Prima di installare il componente aggiuntivo per gli operatori di inferenza tramite la Add-ons console EKS o la CLI, assicurati che siano installate le seguenti dipendenze.

Componenti aggiuntivi EKS richiesti:

- Driver CSI Amazon S3 Mountpoint (versione minima: v1.14.1-eksbuild.1)
- Metrics Server (versione minima: v0.7.2-eksbuild.4)
- Driver Amazon FSx CSI (versione minima: v1.6.0-eksbuild.1)
- Cert Manager (versione minima: v1.18.2-eksbuild.2)

Per istruzioni dettagliate sull'installazione per ciascuna dipendenza, vedere Installazione dell'operatore di inferenza. https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-hyperpod-model-deployment-setup.html

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede le autorizzazioni IAM e utilizza. OIDC/IRSA

Le seguenti policy gestite sono consigliate in quanto forniscono le autorizzazioni minime con ambito di applicazione:

- `AmazonSageMakerHyperPodInferenceAccess`- fornisce i privilegi di amministratore necessari per configurare l'operatore di inferenza
- `AmazonSageMakerHyperPodGatedModelAccess`- fornisce SageMaker HyperPod l'accesso ai modelli controllati in SageMaker Jumpstart (ad esempio, Meta Llama,) GPT-Neo

Per ulteriori informazioni, vedere Installazione dell'operatore di inferenza.

Informazioni aggiuntive

Per saperne di più sull'operatore di SageMaker HyperPod inferenza Amazon, consulta operatore di SageMaker HyperPod inferenza.

Per informazioni sulla risoluzione dei problemi, consulta SageMaker HyperPod Troubleshooting model deployment.

## AWS Network Flow Monitor Agent

Amazon CloudWatch Network Flow Monitor Agent è un'applicazione Kubernetes che raccoglie le statistiche di connessione TCP da tutti i nodi di un cluster e pubblica report sui flussi di rete nelle API di Amazon CloudWatch Network Flow Monitor Ingestion.

Il nome del componente aggiuntivo di Amazon EKS è `aws-network-flow-monitoring-agent`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede autorizzazioni IAM.

Quindi, è necessario aggiungere la policy gestita da `CloudWatchNetworkFlowMonitorAgentPublishPolicy` a tale componente aggiuntivo.

Per ulteriori informazioni sulla configurazione IAM richiesta, consulta IAM Policy nel repository Amazon Network Flow Monitor Agent. CloudWatch GitHub

Per ulteriori informazioni sulla policy gestita, consulta CloudWatchNetworkFlowMonitorAgentPublishPolicy la Amazon CloudWatch User Guide.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta il repository Amazon CloudWatch Network Flow Monitor GitHub Agent.

## Agente di monitoraggio del nodo

Il componente aggiuntivo Amazon EKS dell'agente di monitoraggio dei nodi è in grado di rilevare ulteriori problemi di integrità dei nodi. Questi segnali di integrità aggiuntivi possono essere sfruttati anche dalla funzione opzionale di riparazione automatica dei nodi per sostituire automaticamente i nodi secondo necessità.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `eks-node-monitoring-agent`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione supplementare.

Informazioni aggiuntive

Per ulteriori informazioni, consulta the section called "Salute e riparazione dei nodi".

## AWS Distro per OpenTelemetry

Il componente aggiuntivo AWS Distro for OpenTelemetry Amazon EKS è una distribuzione sicura, pronta per la produzione e supportata del progetto. AWS OpenTelemetry Per ulteriori informazioni, consulta AWS Distro for on. OpenTelemetry GitHub

Il nome del componente aggiuntivo di Amazon EKS è `adot`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede le autorizzazioni IAM solo se si utilizza una delle risorse personalizzate preconfigurate a cui è possibile accedere tramite la configurazione avanzata.

Informazioni aggiuntive

Per ulteriori informazioni, vedi Guida introduttiva a AWS Distro per OpenTelemetry l'utilizzo di EKS Add-Ons nella AWS Distro per la documentazione. OpenTelemetry

ADOT richiede che il `cert-manager` componente aggiuntivo sia distribuito nel cluster come prerequisito, altrimenti questo componente aggiuntivo non funzionerà se distribuito direttamente utilizzando la proprietà. https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest `cluster_addons` Per ulteriori requisiti, consulta Requisiti per iniziare con AWS Distro per OpenTelemetry l'utilizzo di EKS Add-Ons nella AWS distribuzione per OpenTelemetry la documentazione.

## Agente Amazon GuardDuty

Il componente aggiuntivo Amazon GuardDuty Agent Amazon EKS raccoglie gli eventi di runtime (accesso ai file, esecuzione di processi, connessioni di rete) dai nodi del cluster EKS per l'analisi mediante GuardDuty Runtime Monitoring. GuardDuty di per sé (non l'agente) è il servizio di monitoraggio della sicurezza che analizza ed elabora le fonti di dati fondamentali, tra cui eventi di AWS CloudTrail gestione e log di flusso di Amazon VPC, oltre a funzionalità come i log di audit di Kubernetes e il monitoraggio del runtime.

Il nome del componente aggiuntivo di Amazon EKS è `aws-guardduty-agent`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per ulteriori informazioni, consulta Runtime Monitoring for Amazon EKS cluster in Amazon. GuardDuty

- Per rilevare potenziali minacce alla sicurezza nei tuoi cluster Amazon EKS, abilita il monitoraggio del GuardDuty runtime di Amazon e distribuisci l'agente di GuardDuty sicurezza nei tuoi cluster Amazon EKS.

## Agente Amazon Observability CloudWatch

Il componente aggiuntivo Amazon EKS dell'agente Amazon CloudWatch Observability è il servizio di monitoraggio e osservabilità fornito da. AWS Questo componente aggiuntivo installa l' CloudWatch agente e abilita sia CloudWatch Application Signals che CloudWatch Container Insights con una migliore osservabilità per Amazon EKS. Per ulteriori informazioni, consulta Amazon Agent. CloudWatch

Il nome del componente aggiuntivo di Amazon EKS è `amazon-cloudwatch-observability`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called “Credenziali con IRSA”. Le autorizzazioni nelle policy AWSXrayWriteOnlyAccess CloudWatchAgentServerPolicy AWS gestite sono obbligatorie. È possibile creare un ruolo IAM, collegarvi una delle policy gestite e annotare l'account del servizio

Kubernetes utilizzato dal componente aggiuntivo con il comando seguente. Sostituisci *my-cluster* con il nome del cluster e *AmazonEKS_Observability_role* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo, collegarvi la policy e annotare l'account del servizio Kubernetes, consulta the section called "Assegnazione di un ruolo IAM".

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws: iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

Informazioni aggiuntive

Per ulteriori informazioni, vedere Installare l' CloudWatch agente.

## AWS Connettore CA privato per Kubernetes

Il AWS Private CA Connector for Kubernetes è un componente aggiuntivo per cert-manager che consente agli utenti di ottenere certificati da Private Certificate Authority (Private CA). AWS AWS

- Il nome del componente aggiuntivo di Amazon EKS è `aws-privateca-connector-for-kubernetes`.
- Il namespace del componente aggiuntivo è `aws-privateca-issuer`.

Questo componente aggiuntivo richiede `cert-manager`. `cert-manager` è disponibile su Amazon EKS come componente aggiuntivo della community. Per ulteriori informazioni su questo componente aggiuntivo, consulta the section called "Gestione certificati". Per ulteriori informazioni sui componenti aggiuntivi di installazione, consulta the section called "Creare un componente aggiuntivo".

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede autorizzazioni IAM.

Usa EKS Pod Identity per collegare la policy IAM `AWSPrivateCAConnectorForKubernetesPolicy` all'account del servizio Kubernetes `aws-`

`privateca-issuer`. Per ulteriori informazioni, consulta the section called "Utilizzo di Pod Identities".

Per informazioni sulle autorizzazioni richieste, consulta il Managed Policy Reference. AWSPrivateCAConnectorForKubernetesPolicy AWS

Informazioni aggiuntive

Per ulteriori informazioni, consulta il repository AWS Private CA Issuer for GitHub Kubernetes.

Per ulteriori informazioni sulla configurazione del componente aggiuntivo, consulta values.yaml nel repository. https://github.com/cert-manager/aws-privateca-issuer/blob/main/charts/aws-pca-issuer/values.yaml `aws-privateca-issuer` GitHub Verifica che la versione di values.yaml corrisponda alla versione del componente aggiuntivo installato sul cluster.

Questo componente aggiuntivo tollera la macchia utilizzata dalla modalità EKS Auto`CriticalAddonsOnly`. `system` NodePool Per ulteriori informazioni, consulta the section called "Eseguire componenti aggiuntivi critici".

## EKS Pod Identity Agent

Il componente aggiuntivo di Amazon EKS dell'agente Pod Identity fornisce la possibilità di gestire le credenziali per le applicazioni, analogamente al modo in cui i profili dell'istanza Amazon EC2 forniscono le credenziali alle istanze Amazon EC2.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. La modalità automatica EKS di Amazon si integra con Pod Identity EKS. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `eks-pod-identity-agent`.

Autorizzazioni IAM richieste

Il componente aggiuntivo dell'agente Pod Identity di per sé non richiede un ruolo IAM. Utilizza le autorizzazioni del ruolo IAM del nodo Amazon EKS per funzionare, ma non necessita di un ruolo IAM dedicato per il componente aggiuntivo.

Aggiornare le informazioni

Puoi aggiornare solo una versione secondaria alla volta. Ad esempio, se la versione corrente è `1.28.x-eksbuild.y` e desideri aggiornarla a `1.30.x-eksbuild.y`, devi prima eseguire l'aggiornamento a `1.29.x-eksbuild.y` e quindi l'aggiornamento a `1.30.x-eksbuild.y`. Per ulteriori informazioni sull'aggiornamento del componente aggiuntivo, consulta the section called "Aggiornamento di un componente aggiuntivo".

## SR-IOV Network Metrics Exporter

Il componente aggiuntivo SR-IOV Network Metrics Exporter Amazon EKS raccoglie ed espone le metriche sui dispositivi di rete in formato Prometheus. SR-IOV Consente il monitoraggio delle prestazioni di rete sui nodi bare metal EKS. SR-IOV L'esportatore funziona come DaemonSet su nodi con interfacce di SR-IOV-capable rete ed esporta metriche che possono essere analizzate da Prometheus.

> **Note**
>
> Questo componente aggiuntivo richiede nodi con interfacce di rete. SR-IOV-capable

| Proprietà | Valore |
| --- | --- |
| Add-on nome | `sriov-network-metrics-exporter` |
| Namespace | `monitoring` |
| Documentazione | SR-IOV repository Network Metrics Exporter GitHub |
| Nome account del servizio | Nessuno |
| Policy gestite da IAM | Nessuno |
| Autorizzazioni IAM personalizzate | Nessuno |

## AWS Fornitore di driver CSI Secrets Store

Il AWS provider del driver CSI Secrets Store è un componente aggiuntivo che consente di recuperare i AWS segreti da Secrets Manager e i parametri da AWS Systems Manager Parameter Store e di montarli come file nei pod Kubernetes.

Autorizzazioni IAM richieste

L'add-on non richiede autorizzazioni IAM. Tuttavia, i pod applicativi richiederanno le autorizzazioni IAM per recuperare i AWS segreti da Secrets Manager e i parametri da Systems Manager Parameter Store. AWS Dopo aver installato il componente aggiuntivo, l'accesso deve essere configurato tramite IAM Roles for Service Accounts (IRSA) o EKS Pod Identity. Per utilizzare IRSA, consulta la documentazione di configurazione IRSA di Secrets Manager. Per utilizzare EKS Pod Identity, consulta la documentazione di configurazione di Secrets Manager Pod Identity.

AWS suggerisce la policy `AWSSecretsManagerClientReadOnlyAccess` gestita.

Per ulteriori informazioni sulle autorizzazioni richieste, consulta `AWSSecretsManagerClientReadOnlyAccess` il AWS Managed Policy Reference.

Informazioni aggiuntive

Per ulteriori informazioni, consulta il repository GitHub secrets-store-csi-driver-provider-aws.

Per saperne di più sul componente aggiuntivo, consulta la documentazione di Secrets Manager relativa al componente aggiuntivo. AWS

## Amazon Spaces SageMaker

Amazon SageMaker Spaces Add-on offre la possibilità di eseguire IDE e notebook su EKS o cluster. HyperPod-EKS Gli amministratori possono utilizzare la console EKS per installare l'add-on sul proprio cluster e definire configurazioni di spazio predefinite come immagini, risorse di calcolo, archiviazione locale per le impostazioni del notebook (storage aggiuntivo da collegare ai relativi spazi), file system e script di inizializzazione.

Gli sviluppatori di intelligenza artificiale possono usare kubectl per creare, aggiornare ed eliminare spazi. Hanno la flessibilità di utilizzare le configurazioni predefinite fornite dagli amministratori o personalizzare le impostazioni. Gli sviluppatori di intelligenza artificiale possono accedere ai propri spazi su EKS o HyperPod-EKS utilizzando i loro IDE VS Code locali, and/or il browser web che

ospita il loro JupyterLab CodeEditor IDE su un dominio DNS personalizzato configurato dai loro amministratori. Possono anche utilizzare la funzionalità di port forwarding di Kubernetes per accedere agli spazi nei loro browser web.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-spaces`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede autorizzazioni IAM. Per ulteriori informazioni sulla configurazione IAM richiesta, consulta IAM Permissions Setup nella Amazon Developer Guide. SageMaker

Informazioni aggiuntive

Per saperne di più sull'add-on e sulle sue funzionalità, consulta SageMaker AI Notebooks on HyperPod nella Amazon Developer Guide. SageMaker

## Componenti aggiuntivi della community

Puoi utilizzare le AWS API per installare componenti aggiuntivi della community, come Kubernetes Metrics Server. Puoi scegliere di installare componenti aggiuntivi della community come Amazon EKS Add-ons per ridurre la complessità della manutenzione del software su più cluster.

Ad esempio, puoi utilizzare l' AWS API, la CLI o la console di gestione per installare componenti aggiuntivi della community. Puoi installare un componente aggiuntivo della community durante la creazione del cluster.

Gestisci i componenti aggiuntivi della community proprio come Amazon EKS Add-ons esistente. I componenti aggiuntivi della community sono diversi dai componenti aggiuntivi esistenti in quanto hanno un ambito di supporto unico.

> **Note**
>
> L'utilizzo dei componenti aggiuntivi della community è a discrezione dell'utente. Nell'ambito del modello di responsabilità condivisa tra te e te AWS, ci si aspetta che tu comprenda cosa stai installando nel tuo cluster per questi plug-in di terze parti. Sei anche responsabile dei componenti aggiuntivi della community che soddisfano le sue esigenze di sicurezza del cluster. Per ulteriori informazioni, consulta the section called "Supporto per il software distribuito su EKS".

I componenti aggiuntivi della community non sono creati da. AWS AWS convalida i componenti aggiuntivi della community solo per la compatibilità delle versioni. Ad esempio, se installi un componente aggiuntivo della community su un cluster, AWS verifica se è compatibile con la versione Kubernetes del cluster.

È importante sottolineare AWS che non fornisce il supporto completo per i componenti aggiuntivi della community. AWS supporta solo le operazioni del ciclo di vita eseguite utilizzando le AWS API, come l'installazione di componenti aggiuntivi o l'eliminazione di componenti aggiuntivi.

Se è necessario supporto per un componente aggiuntivo della community, utilizzare le risorse di progetto esistenti. Ad esempio, puoi creare un GitHub problema nel repository del progetto.

## Determinare il tipo di componente aggiuntivo

Puoi utilizzare la AWS CLI per determinare il tipo di Amazon EKS. Add-on

Utilizza il seguente comando della CLI per recuperare le informazioni su un componente aggiuntivo. Puoi sostituire `metrics-server` con il nome di qualsiasi componente aggiuntivo.

```
aws eks describe-addon-versions --addon-name metrics-server
```

Esamina l'output della CLI per il campo `owner`.

```
{
    "addons": [
        {
            "addonName": "metrics-server",
            "type": "observability",
            "owner": "community",
            "addonVersions": [
```

Se il valore di `owner` è`community`, allora il componente aggiuntivo è un componente aggiuntivo della community. AWS fornisce solo supporto per l'installazione, l'aggiornamento e la rimozione del componente aggiuntivo. Se hai domande sulla funzionalità e sul funzionamento del componente aggiuntivo stesso, utilizza le risorse della community come GitHub Issues.

## Installare o aggiornare il componente aggiuntivo della community

Puoi installare o aggiornare i componenti aggiuntivi della community allo stesso modo degli altri Amazon EKS Add-ons.

- the section called "Creare un componente aggiuntivo"
- the section called "Aggiornamento di un componente aggiuntivo"
- the section called "Rimozione di un componente aggiuntivo"

## Componenti aggiuntivi della community disponibili

I seguenti componenti aggiuntivi della community sono disponibili presso Amazon EKS.

- the section called "Kubernetes Metrics Server"
- the section called "kube-state-metrics"
- the section called "Prometheus Node Exporter"
- the section called "Gestione certificati"
- the section called "DNS esterno"
- the section called "Fluent Bit"

### Kubernetes Metrics Server

Il Kubernetes Metrics Server è una fonte scalabile ed efficiente di metriche delle risorse dei container per le pipeline di scalabilità automatica integrate in Kubernetes. Raccoglie le metriche delle risorse da Kubelets e le espone nell'apiserver Kubernetes tramite l'API Metriche per essere utilizzate da Horizontal Pod Autoscaler e Vertical Pod Autoscaler.

| Proprietà | Valore |
|---|---|
| Add-on nome | `metrics-server` |
| Namespace | `kube-system` |
| Documentazione | GitHub Readme |
| Nome account del servizio | Nessuno |
| Policy gestite da IAM | Nessuno |
| Autorizzazioni IAM personalizzate | Nessuno |

kube-state-metrics

Add-on agente per generare ed esporre metriche a livello di cluster.

Lo stato degli oggetti di Kubernetes nell'API Kubernetes può essere esposto sotto forma di metriche. Un agente aggiuntivo chiamato kube-state-metrics può connettersi al server API Kubernetes ed esporre un endpoint HTTP con metriche generate dallo stato dei singoli oggetti nel cluster. Espone varie informazioni sullo stato degli oggetti come etichette e annotazioni, tempi di startup e terminazione, stato o Passaggio in cui si trova attualmente l'oggetto.

| Proprietà | Valore |
|---|---|
| Add-on nome | `kube-state-metrics` |
| Namespace | `kube-state-metrics` |
| Documentazione | Metrics for Kubernetes Object States nella documentazione di Kubernetes |
| Nome account del servizio | Nessuno |
| Policy gestite da IAM | Nessuno |
| Autorizzazioni IAM personalizzate | Nessuno |

Prometheus Node Exporter

Esportatore Prometheus per i parametri hardware e del sistema operativo esposti dai kernel *NIX, scritto in Go con collettori di metriche collegabili. Prometheus Node Exporter espone un'ampia varietà di metriche relative all'hardware e al kernel.

| Proprietà | Valore |
|---|---|
| Add-on nome | `prometheus-node-exporter` |
| Namespace | `prometheus-node-exporter` |
| Documentazione | Monitoring Linux host metrics with the Node Exporter nella documentazione di Prometheus |

| Proprietà | Valore |
| --- | --- |
| Nome account del servizio | Nessuno |
| Policy gestite da IAM | Nessuno |
| Autorizzazioni IAM personalizzate | Nessuno |

Gestione certificati

Gestione certificati può essere utilizzato per gestire la creazione e il rinnovo dei certificati.

| Proprietà | Valore |
| --- | --- |
| Add-on nome | `cert-manager` |
| Namespace | `cert-manager` |
| Documentazione | Cert Manager Docs |
| Nome account del servizio | Nessuno |
| Policy gestite da IAM | Nessuno |
| Autorizzazioni IAM personalizzate | Nessuno |

DNS esterno

Il componente aggiuntivo del DNS esterno di EKS può essere utilizzato per gestire i record DNS di Route53 tramite le risorse di Kubernetes.

Le autorizzazioni del DNS esterno possono essere ridotte a `route53:ChangeResourceRecordSets`, `route53:ListHostedZones` e `route53:ListResourceRecordSets` sulle zone ospitate che si desidera gestire.

| Proprietà | Valore |
| --- | --- |
| Add-on nome | `external-dns` |

| Proprietà | Valore |
| --- | --- |
| Namespace | `external-dns` |
| Documentazione | GitHub Readme |
| Nome account del servizio | `external-dns` |
| Policy IAM gestite | `arn:aws: iam::aws:policy/AmazonRoute53FullAccess` |
| Autorizzazioni IAM personalizzate | Nessuno |

Fluent Bit

Fluent Bit è un processore e server d'inoltro di log leggero e ad alte prestazioni. Consente di raccogliere dati data/logs da diverse fonti, unificarli e inviarli a più destinazioni, tra cui Amazon CloudWatch Logs, Amazon S3 e Amazon Data Firehose. Fluent Bit è progettato pensando alle prestazioni e all'efficienza delle risorse, il che lo rende ideale per gli ambienti Kubernetes.

Questo componente aggiuntivo non richiede autorizzazioni IAM nella configurazione predefinita. Tuttavia, potrebbe essere necessario concedere a questo componente aggiuntivo le autorizzazioni IAM se configuri una posizione di output. AWS Per ulteriori informazioni, consulta the section called "Utilizzo di Pod Identities".

| Proprietà | Valore |
| --- | --- |
| Add-on nome | `fluent-bit` |
| Namespace | `fluent-bit` |
| Documentazione | Fluent Bit Documentation |
| Nome account del servizio | `fluent-bit` |
| Policy IAM gestite | Nessuno |
| Autorizzazioni IAM personalizzate | Nessuno |

### Visualizzare le attribuzioni

Puoi scaricare le attribuzioni open source e le informazioni sulle licenze per i componenti aggiuntivi della community.

1. Determina il nome e la versione del componente aggiuntivo per il quale desideri scaricare le attribuzioni.
2. Aggiorna il seguente comando con il nome e la versione:

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/<add-on-name>/<add-on-version>/attributions.zip
   ```

   Esempio:

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/kube-state-metrics/v2.14.0-eksbuild.1/attributions.zip
   ```
3. Utilizza il comando per scaricare il file.

Utilizza questo file zip per visualizzare informazioni sulle attribuzioni delle licenze.

## AWS Componenti aggiuntivi di Marketplace

Oltre all'elenco precedente di componenti aggiuntivi di Amazon EKS, puoi anche aggiungere un'ampia selezione di componenti software aggiuntivi operativi per Amazon EKS offerti da fornitori di software indipendenti. Scegli un componente aggiuntivo per avere maggiori informazioni sul componente e sui relativi requisiti di installazione.

## Accuknox

Il nome del componente aggiuntivo è `accuknox_kubearmor` e il namespace è `kubearmor`. Accuknox pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Getting Started with KubeArmor nella KubeArmor documentazione.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Akuity

Il nome del componente aggiuntivo è `akuity_agent` e il namespace è `akuity`. Akuity pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione dell'agente Akuity su Amazon EKS con il componente aggiuntivo Akuity EKS nella documentazione della piattaforma Akuity.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Calyptia

Il nome del componente aggiuntivo è `calyptia_fluent-bit` e il namespace è `calyptia-fluentbit`. Calyptia pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Guida introduttiva all'agente Calyptia Core sul sito Web della documentazione di Calyptia.

Nome account del servizio

Il nome dell'account del servizio è `calyptia-fluentbit`.

AWS politica IAM gestita

Questo componente aggiuntivo utilizza la policy gestita da `AWSMarketplaceMeteringRegisterUsage`. Per ulteriori informazioni, consulta AWSMarketplaceMeteringRegisterUsage la AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *`my-cluster`* con il nome del cluster e *`my-calyptia-role`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se

devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo IAM".

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

## Observability Collector di Cisco

Il nome del componente aggiuntivo è `cisco_cisco-cloud-observability-collectors` e il namespace è `appdynamics`. Cisco pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Utilizzare Cisco Cloud Observability AWS Marketplace Add-Ons nella documentazione di Cisco. AppDynamics

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS policy IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Cisco Observability Operator

Il nome del componente aggiuntivo è `cisco_cisco-cloud-observability-operators` e il namespace è `appdynamics`. Cisco pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Utilizzare Cisco Cloud Observability AWS Marketplace Add-Ons nella documentazione di Cisco. AppDynamics

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS policy IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## CLOUDSOFT

Il nome del componente aggiuntivo è `cloudsoft_cloudsoft-amp` e il namespace è `cloudsoft-amp`. CLOUDSOFT pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Componente aggiuntivo di Amazon EKS nella documentazione CLOUDSOFT.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Cribl

Il nome del componente aggiuntivo è `cribl_cribledge` e il namespace è `cribledge`. Cribl pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione di Cribl Amazon EKS Add-on for Edge nella documentazione di Cribl

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Dynatrace

Il nome del componente aggiuntivo è `dynatrace_dynatrace-operator` e il namespace è `dynatrace`. Dynatrace pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Monitoraggio di Kubernetes nella documentazione di Dynatrace.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Datree

Il nome del componente aggiuntivo è `datree_engine-pro` e il namespace è `datree`. Datree pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Amazon EKS-integration nella documentazione di Datree.

Nome account del servizio

Il nome dell'account di servizio è datree-webhook-server-awsmp.

AWS politica IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consultare AWSLicenseManagerConsumptionPolicy la AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di

identità OIDC IAM". Sostituisci *my-cluster* con il nome del cluster e *my-datree-role* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo IAM".

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Datadog

Il nome del componente aggiuntivo è `datadog_operator` e il namespace è `datadog-agent`. Datadog pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione dell'agente Datadog su Amazon EKS con l'operatore Datadog nella documentazione di Datadog. Add-on

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Groundcover

Il nome del componente aggiuntivo è `groundcover_agent` e il namespace è`groundcover`. Groundcover pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione di groundcover Amazon EKS Add-on nella documentazione di groundcover.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## IBM Instana

Il nome del componente aggiuntivo è `instana-agent` e il namespace è `instana-agent`. IBM pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Implementare l'osservabilità per i carichi di lavoro Amazon EKS utilizzando il componente aggiuntivo Instana Amazon EKS e Monitora e ottimizza i costi di Amazon EKS con IBM Instana e Kubecost nel blog. AWS

Instana Observability (Instana) offre un Amazon EKS che distribuisce agenti Instana nei cluster Amazon EKS. Add-on I clienti possono utilizzare questo componente aggiuntivo per raccogliere e analizzare dati sulle prestazioni in tempo reale per ottenere approfondimenti sulle loro applicazioni containerizzate. Il componente aggiuntivo Instana Amazon EKS offre visibilità su tutti gli ambienti Kubernetes. Una volta distribuito, l'agente Instana rileva automaticamente i componenti all'interno dei cluster Amazon EKS, inclusi nodi, namespace, implementazioni, servizi e pod.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Grafana Labs

Il nome del componente aggiuntivo è `grafana-labs_kubernetes-monitoring` e il namespace è `monitoring`. Grafana Labs pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Configurare Kubernetes Monitoring as an Add-on with Amazon EKS nella documentazione di Grafana Labs.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Guance

- Publisher: GUANCE
- Nome: `guance_datakit`
- Namespace: `datakit`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: consulta Using Amazon EKS add-on nella documentazione di Guance.

## Proxy HA

Il nome è `haproxy-technologies_kubernetes-ingress-ee` e il namespace è `haproxy-controller`. HA Proxy pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, vedi Installazione AWS nella documentazione di HA Proxy.

Nome account del servizio

Il nome dell'account del servizio è `customer defined`.

AWS politica IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consultare AWSLicenseManagerConsumptionPolicy la AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *my-cluster* con il nome del cluster e *my-haproxy-role* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo IAM".

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Kpow

Il nome del componente aggiuntivo è `factorhouse_kpow` e il namespace è `factorhouse`. Factorhouse pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta AWS Marketplace LM nella documentazione di Kpow.

Nome account del servizio

Il nome dell'account del servizio è `kpow`.

AWS policy IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consultare AWSLicenseManagerConsumptionPolicy la AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *my-cluster* con il nome del cluster e *my-kpow-role* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo IAM".

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Kubecost

Il nome del componente aggiuntivo è `kubecost_kubecost` e il namespace è `kubecost`. Kubecost pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Integrazione di AWS Cloud Billing nella documentazione di Kubecost.

I volumi Store Kubernetes con Amazon EBS devono essere installati nel cluster. Altrimenti riceverai un errore.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Kasten

Il nome del componente aggiuntivo è `kasten_k10` e il namespace è `kasten-io`. Kasten by Veeam pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione di K10 sull' AWS uso di Amazon EKS Add-on nella documentazione di Kasten.

È necessario che il driver CSI di Amazon EBS sia installato nel cluster `StorageClass` predefinito.

Nome account del servizio

Il nome dell'account del servizio è `k10-k10`.

AWS politica IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consultare AWSLicenseManagerConsumptionPolicy la AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *`my-cluster`* con il nome del cluster e *`my-kasten-role`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo IAM".

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-
cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Kong

Il nome del componente aggiuntivo è `kong_konnect-ri` e il namespace è `kong`. Kong pubblica il componente aggiuntivo.

Per informazioni sull'add-on, vedi Installazione di Kong Gateway EKS Add-on nella documentazione di Kong.

I volumi Store Kubernetes con Amazon EBS devono essere installati nel cluster. Altrimenti riceverai un errore.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## LeakSignal

Il nome del componente aggiuntivo è `leaksignal_leakagent` e il namespace è. `leakagent` LeakSignal pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, vedere https://www.leaksignal.com/docs/LeakAgent/Deployment/AWS%20EKS%20Addon/ [Installare il componente aggiuntivo] nella documentazione LeakAgent LeakSignal

I volumi Store Kubernetes con Amazon EBS devono essere installati nel cluster. Altrimenti riceverai un errore.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## NetApp

Il nome del componente aggiuntivo è `netapp_trident-operator` e il namespace è. `trident` NetApp pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, vedere Configurare il componente aggiuntivo Trident EKS nella documentazione. NetApp

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## New Relic

Il nome del componente aggiuntivo è `new-relic_kubernetes-operator` e il namespace è `newrelic`. New Relic pubblica il componente aggiuntivo.

Per informazioni sull'add-on, vedi Installazione di New Relic Add-on for EKS nella documentazione di New Relic.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Rafay

Il nome del componente aggiuntivo è `rafay-systems_rafay-operator` e il namespace è `rafay-system`. Rafay pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione di Rafay Amazon EKS Add-on nella documentazione di Rafay.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Rad Security

- Editore: RAD SECURITY
- Nome: `rad-security_rad-security`
- Namespace: `ksoc`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: vedi Installazione di Rad tramite The AWS Marketplace nella documentazione di Rad Security.

## SolarWinds

- Publisher: SOLARWINDS
- Nome: `solarwinds_swo-k8s-collector-addon`
- Namespace: `solarwinds`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.

- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: consulta Monitorare un cluster Amazon EKS nella SolarWinds documentazione.

## Solo

Il nome del componente aggiuntivo è `solo-io_istio-distro` e il namespace è `istio-system`. Solo pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione di Istio nella Solo.io documentazione.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS policy IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Snyk

- Publisher: SNYK
- Nome: `snyk_runtime-sensor`
- Namespace: `snyk_runtime-sensor`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: consulta Sensore di runtime Snyk nei documenti per gli utenti di Snyk.

## Stormforge

Il nome del componente aggiuntivo è `stormforge_optimize-Live` e il namespace è `stormforge-system`. Stormforge pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, vedere Installazione dell' StormForge agente nella StormForge documentazione.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## SUSE

- Publisher: SUSE
- Nome: `suse_observability-agent`
- Namespace: `suse-observability`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni per la configurazione e l'uso: consulta Avvio rapido nella documentazione di SUSE.

## Splunk

Il nome del componente aggiuntivo è `splunk_splunk-otel-collector-chart` e il namespace è `splunk-monitoring`. Splunk pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installa il componente aggiuntivo di Splunk per Amazon EKS nella documentazione di Splunk.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Teleport

Il nome del componente aggiuntivo è `teleport_teleport` e il namespace è `teleport`. Teleport pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Come funziona Teleport nella documentazione di Teleport.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Tetrate

Il nome del componente aggiuntivo è `tetrate-io_istio-distro` e il namespace è `istio-system`. Tetrate Io pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta il sito Web Tetrate Istio Distro.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Upbound Universal Crossplane

Il nome del componente aggiuntivo è `upbound_universal-crossplane` e il namespace è `upbound-system`. Upbound pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Upbound Universal Crossplane (UXP) nella documentazione di Upbound.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Upwind

Il nome del componente aggiuntivo è `upwind` e il namespace è `upwind`. Upwind pubblica il componente aggiuntivo.

Per ulteriori informazioni sul componente aggiuntivo, consulta Documentazione di Upwind.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

## Creare un componente aggiuntivo Amazon EKS

I componenti aggiuntivi Amazon EKS sono software aggiuntivi per cluster Amazon EKS. Tutti i componenti aggiuntivi di Amazon EKS:

- Includono le patch di sicurezza e le correzioni dei bug più recenti.
- Sono convalidati AWS per funzionare con Amazon EKS.
- Riducono la quantità di lavoro necessaria per gestire il software del componente aggiuntivo.

Puoi creare un componente aggiuntivo Amazon EKS utilizzando `eksctl` Console di gestione AWS, o la CLI AWS . Se il componente aggiuntivo richiede un ruolo IAM, consultare i dettagli del componente aggiuntivo specifico alla pagina Amazon EKS add-ons per conoscere i dettagli sulla creazione del ruolo.

### Prerequisiti

Completare le seguenti operazioni prima di creare un componente aggiuntivo:

- Il cluster deve esistere prima di poter creare un componente aggiuntivo. Per ulteriori informazioni, consulta the section called “Creazione di un cluster ”.
- Verificare se il componente aggiuntivo richiede un ruolo IAM. Per ulteriori informazioni, consulta the section called “Verifica la compatibilità”.
- Verifica che la versione del componente aggiuntivo Amazon EKS sia compatibile con il cluster. Per ulteriori informazioni, consulta the section called “Verifica la compatibilità”.
- Verifica che la versione 0.190.0 o successiva dello strumento da riga di `eksctl` comando sia installata sul tuo computer oppure. AWS CloudShell Per ulteriori informazioni, vedere Installazione sul sito Web `eksctl`.

### Procedura

Puoi creare un componente aggiuntivo Amazon EKS utilizzando`eksctl`, o la CLI Console di gestione AWS. AWS Se il componente aggiuntivo richiede un ruolo IAM, consultare i dettagli del componente aggiuntivo specifico alla pagina Available Amazon EKS add-ons from AWS per conoscere i dettagli sulla creazione del ruolo.

## Creare un componente aggiuntivo (eksctl)

1. Visualizza i nomi dei componenti aggiuntivi disponibili per una versione del cluster. Sostituisci *1.36* con la versione del cluster.

```
eksctl utils describe-addon-versions --kubernetes-version 1.36 | grep AddonName
```

   Di seguito viene riportato un output di esempio:

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
                        "AddonName": "vpc-cni",
                        "AddonName": "adot",
                        "AddonName": "dynatrace_dynatrace-operator",
                        "AddonName": "upbound_universal-crossplane",
                        "AddonName": "teleport_teleport",
                        "AddonName": "factorhouse_kpow",
                        [...]
```

2. Visualizza le versioni disponibili per il componente aggiuntivo da creare. Sostituisci *1.36* con la versione del cluster. Sostituisci *name-of-addon* con il nome del componente aggiuntivo di cui visualizzare le versioni. Il nome deve essere uno di quelli restituiti al passaggio precedente.

```
eksctl utils describe-addon-versions --kubernetes-version 1.36 --name name-of-addon |
 grep AddonVersion
```

   L'output seguente è un esempio di ciò che viene restituito per il componente aggiuntivo denominato `vpc-cni`. Come vedi, il componente aggiuntivo ha diverse versioni disponibili.

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

   a. Determina se il componente aggiuntivo che desideri creare è un componente aggiuntivo di Amazon EKS o AWS Marketplace. Il AWS Marketplace dispone di componenti aggiuntivi di terze parti che richiedono il completamento di passaggi aggiuntivi per creare il componente aggiuntivo.

```
eksctl utils describe-addon-versions --kubernetes-version 1.36 --name name-of-
addon | grep ProductUrl
```

Se non viene restituito alcun output, il componente aggiuntivo è un componente aggiuntivo di Amazon EKS. Se viene restituito un output, il componente aggiuntivo è un AWS componente aggiuntivo di Marketplace. Il seguente output viene fornito per un componente aggiuntivo denominato `teleport_teleport`.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?sku=3bda70bb-566f-4976-806c-
f96faef18b26"
```

Puoi trovare ulteriori informazioni sul componente aggiuntivo nel AWS Marketplace con l'URL restituito. Se il componente aggiuntivo richiede un abbonamento, puoi abbonarti al componente aggiuntivo tramite il Marketplace. AWS Se intendi creare un componente aggiuntivo dal AWS Marketplace, il responsabile IAM che stai utilizzando per creare il componente aggiuntivo deve disporre dell'autorizzazione per creare il ruolo collegato al servizio. AWSServiceRoleForAWSLicenseManagerRole Per ulteriori informazioni sull'assegnazione di autorizzazioni a un'entità IAM, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

3. Crea un componente aggiuntivo di Amazon EKS. Copia il comando e sostituiscilo come segue: *`user-data`*
   - Sostituisci *`my-cluster`* con il nome del cluster.
   - Sostituisci *`name-of-addon`* con il nome del componente aggiuntivo da creare.
   - Se desideri una versione del componente aggiuntivo precedente all'ultima versione, sostituiscila *`latest`* con il numero di versione restituito nell'output di un passaggio precedente che desideri utilizzare.
   - Se il componente aggiuntivo utilizza un ruolo dell'account di servizio, sostituisci *`111122223333`* con l'ID dell'account e sostituisci *`role-name`* con il nome del ruolo. Per istruzioni sulla creazione di un ruolo per l'account di servizio, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Se il componente aggiuntivo non utilizza un ruolo per l'account di servizio, eliminare `--service-account-role-arn arn:aws: iam::111122223333:role/role-name`.

- Questo comando di esempio sovrascrive la configurazione di qualunque versione autogestita esistente del componente aggiuntivo, se presente. Se non desideri sovrascrivere la configurazione di un componente aggiuntivo autogestito esistente, rimuovi l'opzione. *`--force`* Se rimuovi l'opzione e il componente aggiuntivo di Amazon EKS deve sovrascrivere la configurazione di un componente aggiuntivo autogestito esistente, la creazione del componente aggiuntivo di Amazon EKS non riesce e viene visualizzato un messaggio di errore per semplificare la soluzione del conflitto. Prima di specificare questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni da gestire autonomamente, perché questa opzione le sovrascrive.

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest \
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --force
```

Puoi vedere un elenco completo di tutte opzioni disponibili per il comando.

```
eksctl create addon --help
```

Per ulteriori informazioni sulle opzioni disponibili, consulta Componenti aggiuntivi nella documentazione di `eksctl`.

## Crea componente aggiuntivo (AWS Console)

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegliere il nome del cluster per cui si desidera creare il componente aggiuntivo.
4. Scegli la scheda Add-ons.
5. Scegli Ottieni altri componenti aggiuntivi.
6. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster. Puoi aggiungere tutti i componenti aggiuntivi di Amazon EKS e AWS Marketplace di cui hai bisogno.

   Per i componenti aggiuntivi di AWS Marketplace, il responsabile IAM che stai utilizzando per creare il componente aggiuntivo deve disporre delle autorizzazioni per leggere i diritti relativi

al componente aggiuntivo dal. AWS LicenseManager AWS LicenseManager richiede un ruolo AWSServiceRoleForAWSLicenseManagerRole collegato al servizio (SLR) che consenta alle risorse di gestire le licenze per tuo conto. AWS L'SLR è un requisito una tantum per singolo account e non sarà necessario creare SLR separati per ciascun componente aggiuntivo o per ciascun cluster. Per ulteriori informazioni sull'assegnazione di autorizzazioni a un principale IAM, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

Se i componenti aggiuntivi di AWS Marketplace che desideri installare non sono elencati, puoi scegliere la numerazione delle pagine per visualizzare altri risultati di pagina o effettuare una ricerca nella casella di ricerca. Nelle Opzioni di filtro, puoi eseguire una ricerca anche per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca. Dopo aver selezionato i componenti aggiuntivi da installare, scegliere Avanti.

7. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, procedi come segue:
   a. Scegli Visualizza le opzioni di abbonamento per aprire il modulo delle opzioni di abbonamento. Consulta le sezioni Dettagli sui prezzi e Note legali, quindi scegli il pulsante Abbonati per continuare.
   b. Per Versione, selezionare la versione da installare. È preferibile la versione contrassegnata come latest, a meno che il singolo componente aggiuntivo da creare non consigli una versione differente. Per determinare se un componente aggiuntivo ha una versione consigliata, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi".
   c. Sono disponibili due opzioni per configurare i ruoli per i componenti aggiuntivi: il ruolo IAM di EKS Pod Identity e i ruoli IAM per gli account di servizio (IRSA). Seguire la procedura adeguata di seguito per l'opzione preferita. Se tutti i componenti aggiuntivi selezionati includono Richiede un abbonamento in Stato, scegliere Avanti. Non è possibile procedere con la configurazione di questi componenti aggiuntivi finché non sarà stato sottoscritto l'abbonamento dopo la creazione del cluster. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato:
      i. Per il ruolo IAM di Pod Identity per l'account di servizio, è possibile utilizzare un ruolo IAM di EKS Pod Identity esistente o crearne uno con il pulsante Crea ruolo consigliato. Questo campo indicherà solo le opzioni con la policy di attendibilità adeguata. Se non esiste alcun ruolo da selezionare, non si dispone di un ruolo esistente con una policy di attendibilità corrispondente. Per configurare un ruolo IAM di EKS Pod Identity per gli account di servizio del componente aggiuntivo selezionato, scegliere Crea ruolo consigliato. La procedura guidata per la creazione dei ruoli si apre in una finestra separata. La procedura guidata compilerà automaticamente le informazioni sul ruolo come indicato di seguito. Per ogni

componente aggiuntivo in cui si desidera creare il ruolo IAM di EKS Pod Identity, completare i passaggi della procedura guidata IAM come indicato di seguito.

- Nella fase Seleziona entità attendibile, l'opzione di AWS servizio per EKS e il caso d'uso per EKS - Pod Identity sono preselezionate e la politica di fiducia appropriata verrà compilata automaticamente per il componente aggiuntivo. Ad esempio, il ruolo verrà creato con la policy di attendibilità adeguata contenente il principale IAM pods.eks.amazonaws.com, come descritto nella pagina the section called "Vantaggi delle associazioni EKS Pod Identity". Scegli Avanti.
- Al passaggio Aggiungi autorizzazioni, viene preselezionata la policy gestita adeguata per la policy del ruolo per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo CNI di Amazon VPC, il ruolo verrà creato con la policy gestita `AmazonEKS_CNI_Policy`, come descritto nella pagina the section called "Plug-in della CNI di Amazon VPC per Kubernetes". Scegli Avanti.
- Al passaggio Nomina, verifica e crea, in Nome del ruolo, viene inserito automaticamente il nome del ruolo predefinito per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con il nome. AmazonEKSPodIdentityAmazonVPCCNIRole In Descrizione, la descrizione predefinita viene compilata automaticamente con la descrizione adeguata per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con la descrizione Consente ai pod in esecuzione nel cluster Amazon EKS di accedere alle risorse. AWS In Policy di attendibilità, viene visualizzata la policy di attendibilità compilata per il componente aggiuntivo. Scegliere Crea ruolo.

  NOTA: mantenere il nome del ruolo predefinito consente a EKS di preselezionare il ruolo dei componenti aggiuntivi nei nuovi cluster o quando si aggiungono componenti aggiuntivi ai cluster esistenti. È comunque possibile sostituire tale nome e il ruolo sarà disponibile per il componente aggiuntivo in tutti i cluster, ma dovrà essere selezionato manualmente dal menu a discesa.

ii. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato e per i quali si desidera configurare i ruoli utilizzando IRSA, consultare la documentazione relativa al componente aggiuntivo da creare per definire una policy IAM e collegarla a un ruolo. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per selezionare un ruolo IAM è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

iii. Scegli Optional configuration settings (Impostazioni di configurazione facoltative).

iv. Se il componente aggiuntivo richiede una configurazione, inseriscila nella casella Configuration values (Valori di configurazione). Per determinare se il componente aggiuntivo richiede informazioni di configurazione, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi".

v. Selezionare una delle opzioni disponibili per Metodo di risoluzione dei conflitti. Se si sceglie Sostituisci come Metodo di risoluzione dei conflitti, una o più impostazioni per il componente aggiuntivo esistente possono essere sovrascritte con le impostazioni del componente aggiuntivo Amazon EKS. Se non si abilita questa opzione e c'è un conflitto con le impostazioni esistenti, l'operazione non va a buon fine e viene visualizzato un messaggio di errore per aiutarti a risolvere il conflitto. Prima di scegliere questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni che devono essere autogestite.

vi. Se si desidera installare il componente aggiuntivo in un namespace specifico, inserirlo nel campo Namespace. Per AWS i componenti aggiuntivi della community, puoi definire uno spazio dei nomi Kubernetes personalizzato in cui installare il componente aggiuntivo. Per ulteriori informazioni, consulta the section called "Namespace personalizzato per componenti aggiuntivi".

vii. Scegli Avanti.

8. Nella pagina Rivedi e aggiungi, scegli Crea. Una volta completata l'installazione del componente aggiuntivo, vengono visualizzati i componenti aggiuntivi installati.
9. Se uno dei componenti aggiuntivi installati richiede una sottoscrizione, completa la procedura seguente:
   a. scegli il pulsante Subscribe (Sottoscrivi) nell'angolo inferiore destro del componente aggiuntivo. Verrai indirizzato alla pagina del componente aggiuntivo nel Marketplace. AWS Leggi le informazioni sul componente aggiuntivo, ad esempio Product Overview (Panoramica del prodotto) e Pricing information (Informazioni sui prezzi).
   b. Seleziona il pulsante Continue to Subscribe (Continua la sottoscrizione) nella parte superiore destra della pagina del componente aggiuntivo.
   c. Leggi i Terms and Conditions (Termini e condizioni). Se li accetti, scegli Accept Terms (Accetta i termini). L'elaborazione della sottoscrizione può richiedere vari minuti. Durante l'elaborazione della sottoscrizione, il pulsante Return to Amazon EKS Console (Torna alla console Amazon EKS) è disattivato.

d. Al termine dell'elaborazione dell'abbonamento, il pulsante Return to Amazon EKS Console (Torna alla console Amazon EKS) non è più disattivato. Scegli il pulsante per tornare alla Add-ons scheda della console Amazon EKS per il tuo cluster.

e. Per il componente aggiuntivo che hai sottoscritto, scegli Remove and reinstall (Rimuovi e reinstalla), quindi scegli Reinstall add-on (Reinstalla componente aggiuntivo). L'installazione del componente aggiuntivo può richiedere vari minuti. Una volta completata l'installazione, puoi configurare il componente aggiuntivo.

## Crea un componente aggiuntivo (AWS CLI)

1. È necessaria una versione `2.12.3` o successiva o una versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo oppure. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come`yum`,`apt-get`, o Homebrew per macOS sono spesso diverse versioni inferiori all'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata AWS CloudShell potrebbe inoltre essere inferiore a quella più recente. Per aggiornarla, vedi Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
2. Determina quali componenti aggiuntivi sono disponibili. Puoi vedere tutti i componenti aggiuntivi disponibili, incluso il tipo e l'autore. Puoi anche vedere l'URL dei componenti aggiuntivi disponibili nel Marketplace. AWS Sostituisci *`1.36`* con la versione del cluster.

```
aws eks describe-addon-versions --kubernetes-version 1.36 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner, Publisher: publisher, Type: type}' --output table
```

Di seguito viene riportato un output di esempio:

```
-------------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
  |
+-----------------------------------------------------------------
+------------------------------+-----------------+--------------
+---------------------+
|                     MarketplaceProductUrl                     |            Name
          |      Owner       |  Publisher   |        Type         |
```

```
+-------------------------------------------------------------
+------------------------------+-----------------+-------------
+---------------------+
|  None                                                       |  aws-ebs-csi-driver
           |  aws             |  eks         |  storage           |
|  None                                                       |  coredns
           |  aws             |  eks         |  networking        |
|  None                                                       |  kube-proxy
           |  aws             |  eks         |  networking        |
|  None                                                       |  vpc-cni
           |  aws             |  eks         |  networking        |
|  None                                                       |  adot
           |  aws             |  eks         |  observability     |
| https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
 |
| https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |  upbound_universal-
crossplane |  aws-marketplace |  upbound     |  infra-management   |
| https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |  teleport_teleport
          |  aws-marketplace |  teleport    |  policy-management  |
| https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |  factorhouse_kpow
          |  aws-marketplace |  factorhouse |  monitoring         |
|  [...]                                                      |  [...]
           |  [...]           |  [...]       |  [...]             |
+-------------------------------------------------------------
+------------------------------+-----------------+-------------
+---------------------+
```

Il tuo output potrebbe essere diverso. In questo output di esempio, sono disponibili tre diversi componenti aggiuntivi di tipo `networking` e cinque componenti aggiuntivi con un autore di tipo `eks`. I componenti aggiuntivi con `aws-marketplace` nella colonna `Owner` potrebbero richiedere un abbonamento per poter essere installati. Per ulteriori informazioni sul componente aggiuntivo e per sottoscriverlo, puoi visitare l'URL.

3. Puoi vedere quali versioni sono disponibili per ogni componente aggiuntivo. Sostituisci *`1.36`* con la versione del tuo cluster, quindi sostituisci *`vpc-cni`* con il nome del componente aggiuntivo restituito nella fase precedente.

```
aws eks describe-addon-versions --kubernetes-version 1.36 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Di seguito viene riportato un output di esempio:

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

Per impostazione predefinita, la versione con `True` nella colonna `Defaultversion` è la versione con cui viene creato il componente aggiuntivo.

4. (Facoltativo) Trova le opzioni di configurazione del componente aggiuntivo scelto utilizzando il comando seguente:

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.12.0-
eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema\":\"http://
json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":{\"additionalProperties
\":false,\"properties\":{\"hostPath\":{\"$ref\":\"#/definitions/HostPath\"}},\"title
\":\"Cri\",\"type\":\"object\"},\"Env\":{\"additionalProperties\":false,\"properties
\":{\"ADDITIONAL_ENI_TAGS\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":
\"integer\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL\":{\"type\":\"string\"},
\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL
\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION\":{\"format\":\"boolean\",\"type
\":\"string\"},\"DISABLE_METRICS\":{\"format\":\"boolean\",\"type\":\"string
\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING\":{\"format\":\"boolean\",\"type
```

```
\":\"string\"},\"ENABLE_POD_ENI\":{\"format\":\"boolean\",\"type\":\"string
\"},\"ENABLE_PREFIX_DELEGATION\":{\"format\":\"boolean\",\"type\":\"string\"},
\"WARM_ENI_TARGET\":{\"format\":\"integer\",\"type\":\"string\"},\"WARM_PREFIX_TARGET
\":{\"format\":\"integer\",\"type\":\"string\"}},\"title\":\"Env\",\"type\":
\"object\"},\"HostPath\":{\"additionalProperties\":false,\"properties\":{\"path
\":{\"type\":\"string\"}},\"title\":\"HostPath\",\"type\":\"object\"},\"Limits
\":{\"additionalProperties\":false,\"properties\":{\"cpu\":{\"type\":\"string
\"},\"memory\":{\"type\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},
\"Resources\":{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":
\"#/definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},\"title
\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties\":false,
\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":{\"$ref\":\"#/
definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/Resources\"}},\"title\":
\"VpcCni\",\"type\":\"object\"}}}"
}
```

L'output sarà uno schema JSON standard.

Di seguito è riportato un esempio di valori di configurazione validi, in formato JSON, che funzionano con lo schema precedente.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

Di seguito è riportato un esempio di valori di configurazione validi, in formato YAML, che funzionano con lo schema precedente.

```
  resources:
    limits:
      cpu: 100m
```

5. Stabilire se il componente aggiuntivo richiede autorizzazioni IAM. In tal caso, è necessario (1) stabilire se si desidera utilizzare EKS Pod Identity o ruoli IAM per gli account di servizio (IRSA), (2) stabilire l'ARN del ruolo IAM da utilizzare con il componente aggiuntivo e (3) stabilire il nome dell'account di servizio Kubernetes usato dal componente aggiuntivo. Per ulteriori informazioni, consulta the section called "Recupero delle informazioni IAM".

- Amazon EKS suggerisce di utilizzare EKS Pod Identity, se supportato dal componente aggiuntivo. Ciò richiede che l'agente Pod Identity sia installato nel cluster. Per ulteriori informazioni sull'utilizzo di Pod Identities con Add-ons, consulta. the section called "Ruoli IAM"
- Se l'add-on o il tuo cluster non sono configurati per EKS Pod Identities, usa IRSA. Conferma che l'IRSA è impostato sul tuo cluster.
- Consulta la Add-ons documentazione di Amazon EKS per determinare se il componente aggiuntivo richiede autorizzazioni IAM e il nome dell'account di servizio Kubernetes associato.

  a. Crea un componente aggiuntivo di Amazon EKS. Copia il comando seguente sul tuo dispositivo. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

- Sostituisci *my-cluster* con il nome del cluster.
- Sostituisci *vpc-cni* con il nome del componente aggiuntivo da creare restituito nell'output del passaggio precedente.
- Sostituisci *version-number* con il nome del componente aggiuntivo da utilizzare restituito nell'output della fase precedente.
- Per installare il componente aggiuntivo in un namespace Kubernetes personalizzato, aggiungere l'opzione `--namespace-config 'namespace=<my-namespace>'`. Questa opzione è disponibile solo per AWS i componenti aggiuntivi della community. Per ulteriori informazioni, consulta the section called "Namespace personalizzato per componenti aggiuntivi"
- Se il componente aggiuntivo non richiede le autorizzazioni IAM, eliminalo. *<service-account-configuration>*
- Esegui una delle seguenti operazioni:
  - Se l'add-on (1) richiede le autorizzazioni IAM e (2) il cluster utilizza EKS Pod Identities, sostituiscilo *<service-account-configuration>* con la seguente associazione di identità del pod. Sostituiscilo *<service-account-name>* con il nome dell'account di servizio utilizzato dal componente aggiuntivo. Sostituisci *<role-arn>* con l'ARN di un ruolo IAM. Il ruolo deve disporre della policy di attendibilità richiesta da EKS Pod Identity.

    ```
    --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
    ```

  - Se l'add-on (1) richiede le autorizzazioni IAM e (2) il cluster utilizza IRSA, sostituiscilo *<service-account-configuration>* con la seguente configurazione IRSA. Sostituiscilo *111122223333* con l'ID del tuo account e *role-name* con il nome di un ruolo IAM esistente che hai creato. Per istruzioni sulla creazione del ruolo, consultare la documentazione del

componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

```
--service-account-role-arn arn:aws: iam::111122223333:role/role-name
```

- Questi comandi di esempio sovrascrivono l'`--configuration-values`opzione di qualsiasi versione autogestita esistente del componente aggiuntivo, se presente. Sostituiscilo con i valori di configurazione desiderati, come una stringa o un file di input. Se non si desidera fornire valori di configurazione, eliminare l'opzione `--configuration-values`. Se non desideri che la AWS CLI sovrascriva la configurazione di un componente aggiuntivo autogestito esistente, rimuovi l'opzione. *`--resolve-conflicts OVERWRITE`* Se rimuovi l'opzione e il componente aggiuntivo di Amazon EKS deve sovrascrivere la configurazione di un componente aggiuntivo autogestito esistente, la creazione del componente aggiuntivo di Amazon EKS non riesce e viene visualizzato un messaggio di errore per semplificare la soluzione del conflitto. Prima di specificare questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni da gestire autonomamente, perché questa opzione le sovrascrive.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
   <service-account-configuration> --configuration-values 'file://example.yaml' --
resolve-conflicts OVERWRITE
```

Per un elenco completo delle opzioni disponibili, consulta `create-addon` nella Documentazione di riferimento della riga di comando Amazon EKS. Se il componente aggiuntivo creato include `aws-marketplace` elencato nella colonna `Owner` di un passaggio precedente, la creazione potrebbe non riuscire e si potrebbe ricevere un messaggio di errore simile al seguente.

```
{
    "addon": {
```

```
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-on. To
 subscribe, visit the AWS Marketplace console, agree to the seller EULA, select the
 pricing type if required, then re-install the add-on"
                }
            ]
        }
    }
}
```

Se ricevi un errore simile a quello nell'output precedente, visita l'URL nell'output di un passaggio precedente per sottoscrivere il componente aggiuntivo. Dopo la sottoscrizione, esegui nuovamente il comando `create-addon`.

## Aggiornamento di un componente aggiuntivo di Amazon EKS

Amazon EKS non aggiorna automaticamente un componente aggiuntivo quando sono rilasciate nuove versioni o dopo l'aggiornamento del cluster a una nuova versione secondaria di Kubernetes. Per aggiornare un componente aggiuntivo per un cluster esistente, devi avviare l'aggiornamento. Dopo l'avvio dell'aggiornamento, Amazon EKS aggiorna automaticamente il componente aggiuntivo. Prima di aggiornare un componente aggiuntivo, consulta la documentazione attuale del componente aggiuntivo. Per un elenco dei componenti aggiuntivi disponibili, consulta the section called " AWS componenti aggiuntivi". Se il componente aggiuntivo richiede un ruolo IAM, consulta i dettagli del componente aggiuntivo specifico in Componenti aggiuntivi di Available Amazon EKS da AWS per conoscere i dettagli sulla creazione del ruolo.

### Prerequisiti

Completa quanto segue prima di aggiornare un componente aggiuntivo:

- Verificare se il componente aggiuntivo richiede un ruolo IAM. Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi Amazon EKS".

- Verifica che la versione del componente aggiuntivo Amazon EKS sia compatibile con il cluster. Per ulteriori informazioni, consulta the section called "Verifica la compatibilità".

## Procedura

Puoi aggiornare un componente aggiuntivo Amazon EKS utilizzando `eksctl` Console di gestione AWS, o la CLI AWS .

## Aggiornamento del componente aggiuntivo (eksctl)

1. Determina i componenti aggiuntivi e le versioni dei componenti aggiuntivi installati sul tuo cluster. Sostituisci *`my-cluster`* con il nome del cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   Di seguito viene riportato un output di esempio:

   ```
   NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
   coredns     v1.8.7-eksbuild.2    ACTIVE  0
   kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
   vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                v1.12.0-eksbuild.1,v1.11.4-
   eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-eksbuild.1
   ```

   L'output potrebbe avere un aspetto diverso in base ai componenti aggiuntivi e alle versioni presenti nel tuo cluster. Nell'output di esempio precedente, due componenti aggiuntivi esistenti nel cluster hanno versioni più recenti disponibili nella colonna `UPDATE AVAILABLE`.
2. Aggiorna il componente aggiuntivo.
   a. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando in base alla necessità:
      - Sostituisci *`my-cluster`* con il nome del cluster.
      - Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster.
      - Sostituisci *`vpc-cni`* con il nome del componente aggiuntivo da aggiornare restituito nell'output del passaggio precedente.
      - Se desideri aggiornare una versione del componente aggiuntivo precedente all'ultima versione, sostituisci *`latest`* con il numero di versione che vuoi utilizzare e che è stato restituito nell'output di un passaggio precedente. Alcuni componenti aggiuntivi includono versioni consigliate. Per ulteriori informazioni, consulta la documentazione del componente

aggiuntivo che stai aggiornando. Per un elenco dei componenti aggiuntivi, consultathe section called " AWS componenti aggiuntivi". * Se il componente aggiuntivo utilizza un account di servizio Kubernetes e un ruolo IAM, sostituiscilo *111122223333* con l'ID del tuo account e *role-name* con il nome di un ruolo IAM esistente che hai creato. Per istruzioni sulla creazione del ruolo, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Se il componente aggiuntivo non utilizza un account di servizio Kubernetes e un ruolo IAM, elimina la riga `serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name` .

- L'opzione *preserve* mantiene i valori esistenti per il componente aggiuntivo. Se hai impostato valori personalizzati per le impostazioni del componente aggiuntivo e non utilizzi questa opzione, Amazon EKS sovrascrive i tuoi valori con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `overwrite`, tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in `none`, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

b. Esegui il comando modificato per creare il file `update-addon.yaml`.

c. Applica il file di configurazione al tuo cluster.

```
eksctl update addon -f update-addon.yaml
```

Per ulteriori informazioni sull'aggiornamento dei componenti aggiuntivi, consultaAggiornamento dei componenti aggiuntivi nella documentazione di `eksctl`.

## Aggiorna il componente aggiuntivo (AWS Console)

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui aggiornare il componente aggiuntivo.
4. Scegli la scheda Add-ons.
5. Scegli il componente aggiuntivo da aggiornare.
6. Scegli Modifica.
7. Nella *`name of addon`* pagina Configura, procedi come segue:
   a. Scegli Versione da utilizzare. Il componente aggiuntivo potrebbe avere una versione raccomandata. Per ulteriori informazioni, consulta la documentazione del componente aggiuntivo che stai aggiornando. Per un elenco dei componenti aggiuntivi, consulta the section called " AWS componenti aggiuntivi".
   b. Sono disponibili due opzioni per configurare i ruoli per i componenti aggiuntivi: il ruolo IAM di EKS Pod Identity e i ruoli IAM per gli account di servizio (IRSA). Seguire la procedura adeguata di seguito per l'opzione preferita. Se tutti i componenti aggiuntivi selezionati includono Richiede un abbonamento sotto Stato, seleziona Avanti. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato, procedi secondo quanto segue:
      i. Per il ruolo IAM di Pod Identity per l'account di servizio, è possibile utilizzare un ruolo IAM di EKS Pod Identity esistente o crearne uno con il pulsante Crea ruolo consigliato. Questo campo indicherà solo le opzioni con la policy di attendibilità adeguata. Se non esiste alcun ruolo da selezionare, non si dispone di un ruolo esistente con una policy di attendibilità corrispondente. Per configurare un ruolo IAM di EKS Pod Identity per gli account di servizio del componente aggiuntivo selezionato, scegliere Crea ruolo consigliato. La procedura guidata per la creazione dei ruoli si apre in una finestra separata. La procedura guidata compilerà automaticamente le informazioni sul ruolo come indicato di seguito. Per ogni

componente aggiuntivo in cui si desidera creare il ruolo IAM di EKS Pod Identity, completare i passaggi della procedura guidata IAM come indicato di seguito.

- Nella fase Seleziona entità attendibile, l'opzione di AWS servizio per EKS e il caso d'uso per EKS - Pod Identity sono preselezionate e la politica di fiducia appropriata verrà compilata automaticamente per il componente aggiuntivo. Ad esempio, il ruolo verrà creato con la policy di attendibilità adeguata contenente il principale IAM pods.eks.amazonaws.com, come descritto nella pagina the section called "Vantaggi delle associazioni EKS Pod Identity". Scegli Avanti.
- Al passaggio Aggiungi autorizzazioni, viene preselezionata la policy gestita adeguata per la policy del ruolo per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo CNI di Amazon VPC, il ruolo verrà creato con la policy gestita `AmazonEKS_CNI_Policy`, come descritto nella pagina the section called "Plug-in della CNI di Amazon VPC per Kubernetes". Scegli Avanti.
- Al passaggio Nomina, verifica e crea, in Nome del ruolo, viene inserito automaticamente il nome del ruolo predefinito per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con il nome. AmazonEKSPodIdentityAmazonVPCCNIRole In Descrizione, la descrizione predefinita viene compilata automaticamente con la descrizione adeguata per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con la descrizione Consente ai pod in esecuzione nel cluster Amazon EKS di accedere alle risorse. AWS In Policy di attendibilità, viene visualizzata la policy di attendibilità compilata per il componente aggiuntivo. Scegliere Crea ruolo.

  > **Note**
  >
  > Mantenere il nome del ruolo predefinito consente a EKS di preselezionare il ruolo dei componenti aggiuntivi nei nuovi cluster o quando si aggiungono componenti aggiuntivi ai cluster esistenti. È comunque possibile sostituire tale nome e il ruolo sarà disponibile per il componente aggiuntivo in tutti i cluster, ma dovrà essere selezionato manualmente dal menu a discesa.

ii. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato e per i quali si desidera configurare i ruoli utilizzando IRSA, consultare la documentazione relativa al componente aggiuntivo da creare per definire una policy IAM e collegarla a un ruolo. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per selezionare un ruolo IAM è necessario disporre di un gestore

OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

c. Scegliere Impostazioni di configurazione facoltative.

d. In Valori di configurazione, inserisci qualsiasi informazione di configurazione specifica del componente aggiuntivo. Per ulteriori informazioni, consulta la documentazione del componente aggiuntivo che stai aggiornando. Per un elenco di componenti aggiuntivi, consulta the section called " AWS componenti aggiuntivi"... per Metodo di risoluzione dei conflitti, seleziona una delle opzioni disponibili. Se hai impostato valori personalizzati per le impostazioni del componente aggiuntivo, è preferibile l'opzione Preserve (Conserva). Se non scegli questa opzione, Amazon EKS sovrascrive i tuoi valori con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in sovrascrivi, tutte le impostazioni sono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in nessuno, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.

8. Scegli Save changes (Salva modifiche).

## Aggiorna componente aggiuntivo (AWS CLI)

1. È necessaria una versione `2.12.3` o successiva o una versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo oppure. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come`yum`,`apt-get`, o Homebrew per macOS sono spesso diverse versioni inferiori all'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata AWS CloudShell potrebbe inoltre essere inferiore a quella più recente. Per aggiornarla, vedi Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
2. Visualizza l'elenco dei componenti aggiuntivi installati. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks list-addons --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio:

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni"
    ]
}
```

3. Visualizza la versione corrente del componente aggiuntivo da aggiornare. Sostituisci *my-cluster* con il nome del tuo cluster e *vpc-cni* con il nome del componente aggiuntivo da aggiornare.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

Di seguito viene riportato un output di esempio.

```
v1.10.4-eksbuild.1
```

4. Determina le versioni del componente aggiuntivo disponibili per la versione del cluster. Sostituiscilo *1.36* con la versione del tuo cluster e *vpc-cni* con il nome del componente aggiuntivo che desideri aggiornare.

```
aws eks describe-addon-versions --kubernetes-version 1.36 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Di seguito viene riportato un output di esempio.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

Per impostazione predefinita, la versione con `True` nella colonna `Defaultversion` è la versione con cui viene creato il componente aggiuntivo.

5. Aggiorna il componente aggiuntivo. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato. Per ulteriori informazioni su questo comando, consulta update-addon nel Riferimento della riga di comando Amazon EKS.

   - Sostituisci *`my-cluster`* con il nome del cluster.
   - Sostituisci *`vpc-cni`* con il nome del componente aggiuntivo da aggiornare che è stato restituito nell'output di un passaggio precedente.
   - Sostituisci *`version-number`* con la versione a cui eseguire l'aggiornamento restituita nell'output del passaggio precedente. Alcuni componenti aggiuntivi includono versioni consigliate. Per ulteriori informazioni, consulta la documentazione del componente aggiuntivo che stai aggiornando. Per un elenco dei componenti aggiuntivi, consulta. the section called " AWS componenti aggiuntivi" * Se il componente aggiuntivo utilizza un account di servizio Kubernetes e un ruolo IAM, sostituiscilo *`111122223333`* con l'ID del tuo account e *`role-name`* con il nome di un ruolo IAM esistente che hai creato. Per istruzioni sulla creazione del ruolo, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

     Se il componente aggiuntivo non utilizza un account di servizio Kubernetes e un ruolo IAM, elimina la riga `serviceAccountRoleARN: arn:aws: iam::`*`111122223333`*`:role/`*`role-name`* .
   - L'opzione `--resolve-conflicts PRESERVE` mantiene i valori esistenti per il componente aggiuntivo. Se hai impostato valori personalizzati per le impostazioni del componente aggiuntivo e non utilizzi questa opzione, Amazon EKS sovrascrive i tuoi valori con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `OVERWRITE`, tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in `NONE`, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento

potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.

- Se desideri rimuovere tutte le configurazioni personalizzate, esegui l'aggiornamento utilizzando l'opzione *`--configuration-values '{}'`*. In tal modo, verranno ripristinati i valori predefiniti di tutte le configurazioni personalizzate. Se non desideri modificare la configurazione personalizzata, non fornire il flag. *`--configuration-values`* Per modificare una configurazione personalizzata, sostituisci *`{}`* con i nuovi parametri.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

6. Controlla lo stato dell'aggiornamento. Sostituiscilo *`my-cluster`* con il nome del tuo cluster e *`vpc-cni`* con il nome del componente aggiuntivo che stai aggiornando.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Di seguito viene riportato un output di esempio.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
    }
}
```

L'aggiornamento è completo quando lo stato è `ACTIVE`.

## Verifica la compatibilità delle versioni aggiuntive di Amazon EKS con un cluster

Prima di creare un componente aggiuntivo di Amazon EKS, devi verificare che la versione del componente aggiuntivo di Amazon EKS sia compatibile con il cluster.

Utilizza describe-addon-versions API per elencare le versioni disponibili dei componenti aggiuntivi di EKS e quali versioni di Kubernetes supporta ciascuna versione del componente aggiuntivo.

1. Verifica che la AWS CLI sia installata e funzionante. `aws sts get-caller-identity` Se questo comando non funziona, scopri come iniziare a usare la AWS CLI.
2. Determina il nome del componente aggiuntivo per il quale recuperare le informazioni sulla compatibilità della versione, ad esempio `amazon-cloudwatch-observability`.
3. Determina la versione di Kubernetes per il cluster, ad esempio `1.36`.
4. Usa la AWS CLI per recuperare le versioni del componente aggiuntivo compatibili con la versione Kubernetes del tuo cluster.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.36
```

Di seguito viene riportato un output di esempio.

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "vX.X.X-eksbuild.X",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "computeTypes": [
                        "ec2",
                        "auto",
                        "hybrid"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.36",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                }
            ]
```

```
        }
    ]
}
```

Questo output mostra che la versione `vX.X.X-eksbuild.X` del componente aggiuntivo è compatibile con quella `1.36` del cluster di Kubernetes.

### Add-on compatibilità con i tipi di elaborazione

Il `computeTypes` campo nell'`describe-addon-versions`output indica la compatibilità di un componente aggiuntivo con EKS Auto Mode Managed Nodes o Hybrid Nodes. Add-ons i contrassegnati `auto` funzionano con l'infrastruttura AWS gestita basata sul cloud di EKS Auto Mode, mentre quelli contrassegnati `hybrid` possono essere eseguiti su nodi locali collegati al piano di controllo cloud EKS.

Per ulteriori informazioni, consulta the section called “Considerazioni per la modalità automatica di Amazon EKS”.

## Rimozione di un componente aggiuntivo Amazon EKS da un cluster

Puoi rimuovere un componente aggiuntivo Amazon EKS dal tuo cluster utilizzando `eksctl` Console di gestione AWS, o la AWS CLI.

Quando rimuovi un componente aggiuntivo Amazon EKS da un cluster:

- non sono previste interruzioni per la funzionalità fornita dal componente aggiuntivo.
- Se utilizzi ruoli IAM per account di servizio (IRSA) e al componente aggiuntivo è associato un ruolo IAM, il ruolo IAM non è rimosso.
- Se utilizzi Pod Identity, tutte le associazioni Pod Identity di proprietà del componente aggiuntivo sono rimosse. Se si specifica l'`--preserve`opzione per la AWS CLI, le associazioni vengono mantenute.
- Amazon EKS interrompe la gestione delle impostazioni per il componente aggiuntivo.
- La console smette di segnalarti la disponibilità di nuove versioni.
- Non è possibile aggiornare il componente aggiuntivo utilizzando AWS strumenti o API.
- Puoi scegliere di lasciare il software del componente aggiuntivo nel tuo cluster in modo da poterlo gestire autonomamente, oppure puoi rimuoverlo. È necessario rimuovere il software aggiuntivo

dal cluster solo se nessuna risorsa del cluster dipende dalla funzionalità fornita dal componente aggiuntivo.

## Prerequisiti

Completa quanto segue prima di rimuovere un componente aggiuntivo:

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Verificare se il componente aggiuntivo richiede un ruolo IAM. Per ulteriori informazioni, consulta the section called "Recupero delle informazioni IAM".
- La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

## Procedura

Sono possibili due opzioni per rimuovere un componente aggiuntivo Amazon EKS:

- Mantenere il software aggiuntivo sul cluster – Questa opzione rimuove la gestione di Amazon EKS su qualsiasi impostazione. Inoltre, rimuove la possibilità per Amazon EKS di notificare gli aggiornamenti e aggiornare automaticamente il componente aggiuntivo Amazon EKS dopo l'avvio di un aggiornamento. Tuttavia, mantiene il software aggiuntivo sul cluster. Questa opzione rende il componente aggiuntivo come autogestito anziché come un componente aggiuntivo Amazon EKS. Con questa opzione, non ci sono tempi di inattività per il componente aggiuntivo.
- Rimuovere completamente il software aggiuntivo dal cluster – Consigliamo di rimuovere il componente aggiuntivo Amazon EKS dal cluster solo se non ci sono risorse nel cluster che dipendono dallo stesso.

Puoi rimuovere un componente aggiuntivo Amazon EKS utilizzando `eksctl` Console di gestione AWS, o la CLI AWS .

Rimuovere un componente aggiuntivo (eksctl)

1. Determina i componenti aggiuntivi installati sul tuo cluster. Sostituisci *`my-cluster`* con il nome del cluster.

```
eksctl get addon --cluster my-cluster
```

Di seguito viene riportato un output di esempio:

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
[...]
```

L'output potrebbe avere un aspetto diverso in base ai componenti aggiuntivi e alle versioni presenti nel tuo cluster.

2. Rimuovi il componente aggiuntivo. Sostituisci *my-cluster* con il nome del cluster e *name-of-add-on* con il nome del componente aggiuntivo da rimuovere restituito nell'output del passaggio precedente. Se rimuovi l'*--preserve*opzione, oltre al fatto che Amazon EKS non gestisce più il componente aggiuntivo, il software aggiuntivo viene eliminato dal cluster.

```
eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
```

Per ulteriori informazioni sulla rimozione dei componenti aggiuntivi, consulta Eliminare componenti aggiuntivi nella documentazione di `eksctl`.

Rimuovi il componente aggiuntivo (AWS Console)

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui desideri rimuovere il componente aggiuntivo di Amazon EBS.
4. Scegli la scheda Add-ons.
5. Scegli il componente aggiuntivo che vuoi rimuovere.
6. Scegli Rimuovi.
7. Nella finestra di dialogo  Rimuovi: ***name of addon*** conferma, effettuate le seguenti operazioni:
   a. Se desideri che Amazon EKS interrompa la gestione delle impostazioni per il componente aggiuntivo, seleziona Mantieni sul cluster. Effettuare questa operazione se si desidera conservare il software aggiuntivo sul cluster. In questo modo sarà possibile gestire autonomamente tutte le impostazioni del componente aggiuntivo.
   b. Inserisci il nome del componente aggiuntivo.

c. Scegli Rimuovi.

Rimuovi componente aggiuntivo (AWS CLI)

1. È necessaria la versione `0.215.0` o successiva dello strumento da riga di `eksctl` comando installato sul dispositivo oppure AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Visualizza l'elenco dei componenti aggiuntivi installati. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks list-addons --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

3. Rimuovi il componente aggiuntivo installato. Sostituisci *`my-cluster`* con il nome del tuo cluster e *`name-of-add-on`* con il nome del componente aggiuntivo da rimuovere. La rimozione *`--preserve`* comporta l'eliminazione del software aggiuntivo dal cluster.

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --preserve
```

Di seguito è riportato un output di esempio abbreviato.

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
    }
}
```

4. Controlla lo stato della rimozione. Sostituiscilo *my-cluster* con il nome del cluster e *name-of-addon* con il nome del componente aggiuntivo che stai rimuovendo.

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

Di seguito è riportato l'output esemplificativo dopo la rimozione del componente aggiuntivo.

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

## Ruoli IAM per i componenti aggiuntivi di Amazon EKS

Alcuni componenti aggiuntivi di Amazon EKS richiedono ruoli e autorizzazioni IAM per chiamare le API. AWS Ad esempio, il componente aggiuntivo Amazon VPC CNI richiama determinate AWS API per configurare le risorse di rete nel tuo account. A questi componenti aggiuntivi deve essere concessa l'autorizzazione tramite IAM. Più specificamente, l'account di servizio del pod che esegue il componente aggiuntivo deve essere associato a un ruolo IAM con una policy IAM specifica.

Il modo consigliato per concedere le AWS autorizzazioni ai carichi di lavoro del cluster è utilizzare la funzionalità Pod Identities di Amazon EKS. Puoi utilizzare un'associazione di Pod Identity per mappare l'account di servizio di un componente aggiuntivo a un ruolo IAM. Se un pod utilizza un account di servizio con un'associazione, Amazon EKS imposta le variabili di ambiente nei container del pod. Le variabili di ambiente configurano gli AWS SDK, inclusa la AWS CLI, per utilizzare le credenziali EKS Pod Identity. Per ulteriori informazioni, consulta the section called "Pod Identity"

I componenti aggiuntivi di Amazon EKS possono aiutare a gestire il ciclo di vita delle associazioni di Pod Identity corrispondenti al componente aggiuntivo. Ad esempio, puoi creare o aggiornare un componente aggiuntivo di Amazon EKS e l'associazione di Pod Identity necessaria in una singola chiamata API. Amazon EKS fornisce anche un'API per il recupero delle policy IAM suggerite.

1. Verifica che l'agente di identità del pod Amazon EKS sia configurato sul tuo cluster.
2. Determina se il componente aggiuntivo che desideri installare richiede autorizzazioni IAM utilizzando l'operazione `describe-addon-versions` AWS CLI. Se il flag `requiresIamPermissions` è `true`, allora dovresti usare l'operazione `describe-addon-configurations` per determinare le autorizzazioni necessarie al componente aggiuntivo. La risposta include un elenco di policy IAM gestite suggerite.

3. Recupera il nome dell'account di servizio Kubernetes e la policy IAM utilizzando l'operazione `describe-addon-configuration` CLI. Valuta l'ambito della policy suggerita rispetto ai requisiti di sicurezza.
4. Crea un ruolo IAM utilizzando la policy di autorizzazioni suggerita e la policy di attendibilità richiesta da Pod Identity. Per ulteriori informazioni, consulta the section called "Crea un'associazione Pod Identity (AWS Console)".
5. Crea o aggiorna un componente aggiuntivo Amazon EKS utilizzando la CLI. Specifica almeno un'associazione di Pod Identity. Un'associazione di Pod Identity è il nome di un account di servizio Kubernetes e l'ARN del ruolo IAM.
   - Le associazioni di Pod Identity create utilizzando le API del componente aggiuntivo sono di proprietà del rispettivo componente aggiuntivo. Se elimini il componente aggiuntivo, viene eliminata anche l'associazione di Pod Identity. Puoi evitare questa eliminazione a catena utilizzando l'`preserve`opzione quando elimini un componente aggiuntivo utilizzando la CLI o l'API. AWS Puoi anche aggiornare o eliminare direttamente l'associazione dell'identità del pod, se necessario. Add-ons non può assumere la proprietà delle associazioni di identità dei pod esistenti. Devi eliminare l'associazione esistente e ricrearla usando un'operazione di creazione o aggiornamento di un componente aggiuntivo.
   - Amazon EKS consiglia di utilizzare le associazioni di Pod Identity per gestire le autorizzazioni IAM per i componenti aggiuntivi. Il metodo precedente, ruoli IAM per account di servizio (IRSA), è ancora supportato. Puoi specificare sia un `serviceAccountRoleArn` IRSA che un'associazione di Pod Identity per un componente aggiuntivo. Se l'agente Pod Identity EKS è installato nel cluster, l'`serviceAccountRoleArn` verrà ignorato ed EKS utilizzerà l'associazione di Pod Identity fornita. Se Pod Identity non è abilitato, verrà utilizzato l'`serviceAccountRoleArn`.
   - Se aggiorni le associazioni di Pod Identity per un componente aggiuntivo esistente, Amazon EKS avvia un riavvio in sequenza dei pod del componente aggiuntivo.

## Recupero delle informazioni IAM su un componente aggiuntivo di Amazon EKS

Prima di creare un componente aggiuntivo, utilizza la AWS CLI per determinare:

- Se il componente aggiuntivo richiede le autorizzazioni IAM
- La policy IAM consigliata da utilizzare

Procedura

1. Determina il nome del componente aggiuntivo che desideri installare e la versione Kubernetes del tuo cluster. Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".
2. Utilizza la AWS CLI per determinare se il componente aggiuntivo richiede le autorizzazioni IAM.

```
aws eks describe-addon-versions \
--addon-name <addon-name> \
--kubernetes-version <kubernetes-version>
```

Esempio:

```
aws eks describe-addon-versions \
--addon-name aws-ebs-csi-driver \
--kubernetes-version 1.30
```

Esamina il seguente output esemplificativo. Tieni presente che `requiresIamPermissions` è `true` e la versione del componente aggiuntivo predefinita. È necessario specificare la versione del componente aggiuntivo quando si recupera la policy IAM consigliata.

```
{
    "addons": [
        {
            "addonName": "aws-ebs-csi-driver",
            "type": "storage",
            "addonVersions": [
                {
                    "addonVersion": "v1.31.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
```

```
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. Se il componente aggiuntivo richiede le autorizzazioni IAM, utilizza la AWS CLI per recuperare una policy IAM consigliata.

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

Esempio:

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

Esamina l'output successivo. Prendi nota di `recommendedManagedPolicies`.

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicyV2"
        ]
    }
]
```

4. Crea un ruolo IAM e collega la policy gestita consigliata. In alternativa, riconsulta la policy gestita e definisci le autorizzazioni in modo appropriato. Per ulteriori informazioni, consulta the section called "Crea un'associazione Pod Identity (AWS Console)".

Riferimento di supporto Pod Identity

La tabella seguente indica se determinati componenti aggiuntivi di Amazon EKS supportano EKS Pod Identity.

| Nome del componente aggiuntivo | Supporto di Pod Identity | Versione minima richiesta |
|---|---|---|
| Driver CSI per Amazon EBS | Sì | v1.26.0-eksbuild.1 |
| CNI di Amazon VPC | Sì | v1.15.5-eksbuild.1 |
| Driver CSI per Amazon EFS | Sì | v2.0.5-eksbuild.1 |
| AWS Distro per OpenTelemetry | Sì | v0.94.1-eksbuild.1 |
| Driver CSI di Mountpoint per Amazon S3 | No | N/D |
| Agente Amazon CloudWatch Observability | Sì | v3.1.0-eksbuild.1 |

L'ultimo aggiornamento di questa tabella risale al 28 ottobre 2024.

## Utilizzo di Pod Identity per assegnare un ruolo IAM a un componente aggiuntivo Amazon EKS

Alcuni componenti aggiuntivi Amazon EKS richiedono ruoli IAM e autorizzazioni. Prima di aggiornare un componente aggiuntivo Amazon EKS per utilizzare un'associazione Pod Identity, verifica il ruolo e la policy da utilizzare. Per ulteriori informazioni, consulta the section called "Recupero delle informazioni IAM".

1. Determina:
   - `cluster-name`: il nome del cluster su cui installare il componente aggiuntivo.
   - `addon-name`: il nome del componente aggiuntivo da installare.
   - `service-account-name`: il nome dell'account di servizio Kubernetes utilizzato dal componente aggiuntivo.
   - `iam-role-arn`: l'ARN di un ruolo IAM con le autorizzazioni sufficienti per il componente aggiuntivo. Il ruolo deve disporre della policy di attendibilità richiesta per EKS Pod Identity. Per ulteriori informazioni, consulta the section called "Crea un'associazione Pod Identity (AWS Console)".

2. Aggiorna il componente aggiuntivo utilizzando l'interfaccia a riga di comando AWS . È anche possibile specificare le associazioni Pod Identity durante la creazione di un componente aggiuntivo, utilizzando la stessa sintassi `--pod-identity-associations`. Nota che quando specifichi le associazioni di Pod Identity durante l'aggiornamento di un componente aggiuntivo, tutte le associazioni di Pod Identity precedenti sono sovrascritte.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations '[{"serviceAccount": "<service-account-name>", "roleArn":
 "<role-arn>"}]'
```

   Ad esempio:

```
aws eks update-addon --cluster-name mycluster \
--addon-name aws-ebs-csi-driver \
--pod-identity-associations '[{"serviceAccount": "ebs-csi-controller-sa", "roleArn":
 "arn:aws: iam::123456789012:role/StorageDriver"}]'
```

3. Convalida che l'associazione Pod Identity sia stata creata:

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

   A operazione riuscita, viene visualizzato un output simile al seguente. Nota OwnerARN del componente aggiuntivo EKS.

```
{
    "associations": [
        {
            "clusterName": "mycluster",
            "namespace": "kube-system",
            "serviceAccount": "ebs-csi-controller-sa",
            "associationArn": "arn:aws: eks:us-
west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
            "ownerArn": "arn:aws: eks:us-west-2:123456789012:addon/mycluster/aws-ebs-
csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

## Rimozione delle associazioni Pod Identity da un componente aggiuntivo Amazon EKS

Rimuovi le associazioni Pod Identity da un componente aggiuntivo Amazon EKS.

1. Determina:
   - `cluster-name`: il nome del cluster EKS su cui installare il componente aggiuntivo.
   - `addon-name`: Il nome del componente aggiuntivo Amazon EKS da installare.
2. Aggiorna il componente aggiuntivo per specificare un array vuoto di associazioni di Pod Identity.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

## Risoluzione dei problemi in Pod Identity per i componenti aggiuntivi di EKS

Se i componenti aggiuntivi riscontrano errori durante il tentativo di eseguire operazioni su AWS API, SDK o CLI, conferma quanto segue:

- L'agente Pod Identity è installato nel cluster.
  - Per informazioni su come installare l'agente Pod Identity, consulta the section called "Configurazione dell'agente".
- Add-on Ha un'associazione Pod Identity valida.
  - Utilizza la AWS CLI per recuperare le associazioni per il nome dell'account di servizio utilizzato dal componente aggiuntivo.

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- Il ruolo IAM dispone della policy di attendibilità richiesta per Pod Identity.
  - Utilizza la AWS CLI per recuperare la politica di attendibilità per un componente aggiuntivo.

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- Il ruolo IAM dispone delle autorizzazioni necessarie per il componente aggiuntivo.
  - Usalo AWS CloudTrail per rivedere i nostri eventi`AccessDenied`. `UnauthorizedOperation`
- Il nome dell'account di servizio nell'associazione di Pod Identity corrisponde al nome dell'account di servizio utilizzato dal componente aggiuntivo.

  - Per informazioni sui componenti aggiuntivi disponibili, consulta the section called " AWS componenti aggiuntivi".
- Controlla la configurazione di MutatingWebhookConfiguration named `pod-identity-webhook`
  - `admissionReviewVersions` del webhook deve essere `v1beta1` e non funziona con `v1`.

# Determina i campi che puoi personalizzare per i componenti aggiuntivi Amazon EKS

I componenti aggiuntivi Amazon EKS vengono installati nel cluster utilizzando configurazioni standard e best practice. Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Componenti aggiuntivi Amazon EKS".

Potresti voler personalizzare la configurazione di un componente aggiuntivo Amazon EKS per abilitare funzionalità avanzate. Amazon EKS utilizza la funzionalità applicazione lato server Kubernetes per abilitare la gestione di un componente aggiuntivo da parte di Amazon EKS, senza sovrascrivere la configurazione per le impostazioni non gestite da Amazon EKS. Per ulteriori informazioni, consulta Server-Side Apply nella documentazione di Kubernetes. Per raggiungere questo obiettivo, Amazon EKS gestisce un insieme minimo di campi per ogni componente aggiuntivo che installa. Puoi modificare senza problemi tutti i campi non gestiti da Amazon EKS o da un altro processo del piano di controllo Kubernetes, come `kube-controller-manager`.

> **Important**
>
> La modifica di un campo gestito da Amazon EKS impedisce ad Amazon EKS di gestire il componente aggiuntivo e può comportare la sovrascrittura delle modifiche quando un componente aggiuntivo viene aggiornato.

## Sintassi della gestione dei campi

Quando si visualizzano i dettagli di un oggetto Kubernetes, nell'output vengono restituiti sia i campi gestiti che quelli non gestiti. I campi gestiti possono essere dei seguenti tipi:

- Completamente gestito – Tutte le chiavi del campo sono gestite da Amazon EKS. Le modifiche a qualsiasi valore causano un conflitto.
- Parzialmente gestiti – Alcune chiavi per il campo sono gestite da Amazon EKS. Solo le modifiche alle chiavi gestite esplicitamente da Amazon EKS causano un conflitto.

Entrambi i tipi di campi sono taggati con `manager: eks`.

Ogni chiave è un `.` che rappresenta il campo stesso, che viene sempre mappato a un insieme vuoto o a una stringa che rappresenta un sottocampo o un elemento. L'output per la gestione del campo è costituito dai seguenti tipi di dichiarazioni:

- `f:`*`name`* , dove *`name`* è il nome di un campo in un elenco.
- `k:`*`keys`* , *`keys`* dov'è una mappa dei campi di un elemento dell'elenco.
- `v:`*`value`* , dove *`value`* è il valore formattato JSON esatto di una elemento in elenco.
- `i:`*`index`* , *`index`* dov'è la posizione di un elemento nell'elenco.

Le seguenti porzioni di output per il componente aggiuntivo CoreDNS illustrano le dichiarazioni precedenti:

- Campi completamente gestiti – Se un campo gestito ha un `f:` (campo) specificato, ma non `k:` (chiave), l'intero campo è gestito. Le modifiche apportate a qualsiasi valore in questo campo causano un conflitto.

  Nel seguente output, è possibile vedere che il container denominato `coredns` è gestito da `eks`. La `args`, `image`, e i sottocampi `imagePullPolicy` sono gestiti anche da `eks`. Le modifiche a qualsiasi valore in questi campi causano un conflitto.

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
    .: {}
    f:args: {}
    f:image: {}
    f:imagePullPolicy: {}
  [...]
  manager: eks
  [...]
  ```

- Campi parzialmente gestiti – Se una chiave gestita ha un valore specificato, le chiavi dichiarate vengono gestite per quel campo. La modifica delle chiavi specificate causa un conflitto.

  Nel seguente output, è possibile vedere che `eks` gestisce il `config-volume` e volumi `tmp` impostati con la chiave `name`.

```
[...]
f:volumes:
  k:{"name":"config-volume"}:
    .: {}
    f:configMap:
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"tmp"}:
    .: {}
    f:name: {}
[...]
manager: eks
[...]
```

- Aggiunta di chiavi a campi parzialmente gestiti – Se viene gestita solo una chiave-valore specifica, è possibile aggiungere ulteriori chiavi, ad esempio, argomenti, a un campo senza causare conflitti. Se aggiungi altre chiavi, assicurati per prima cosa che il campo non sia gestito. L'aggiunta o la modifica di qualsiasi valore gestito causa un conflitto.

  Nel seguente output, è possibile vedere come sia la chiave `name` che il campo `name` siano gestiti. L'aggiunta o la modifica di qualsiasi nome container causa un conflitto con questa chiave gestita.

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

## Procedura

È possibile utilizzare `kubectl` per vedere quali campi sono gestiti da Amazon EKS per qualsiasi componente aggiuntivo Amazon EKS.

Puoi modificare senza problemi tutti i campi non gestiti da Amazon EKS o da un altro processo del piano di controllo Kubernetes, come `kube-controller-manager`.

1. Determinare quale componente aggiuntivo si desidera esaminare. Per visualizzare tutti i file `deployments` e DaemonSets distribuiti nel cluster, consultathe section called "Accesso alle risorse di cluster".
2. Visualizzazione dei campi gestiti per un componente aggiuntivo eseguendo il comando seguente:

   ```
   kubectl get type/add-on-name -n add-on-namespace -o yaml
   ```

   Ad esempio, è possibile visualizzare i campi gestiti per il componente aggiuntivo CoreDNS con il comando seguente.

   ```
   kubectl get deployment/coredns -n kube-system -o yaml
   ```

   La gestione dei campi è elencata nella sezione seguente dell'output restituito.

   ```
   [...]
   managedFields:
     - apiVersion: apps/v1
       fieldsType: FieldsV1
       fieldsV1:
   [...]
   ```

   > **Note**
   >
   > Se non visualizzi `managedFields` nell'output, aggiungi `--show-managed-fields` al comando ed eseguilo di nuovo. La versione di `kubectl` che stai utilizzando determina se i campi gestiti vengono restituiti per impostazione predefinita.

### Fasi successive

Personalizza i campi non di proprietà del AWS tuo componente aggiuntivo.

## Convalida le firme delle immagini dei container durante l'implementazione

Se utilizzi AWS Signer e desideri verificare le immagini dei container firmate al momento dell'implementazione, puoi utilizzare una delle seguenti soluzioni:

- Gatekeeper e Ratify: usa Gatekeeper come controller di ammissione e Ratify configurato con un plug-in AWS Signer come Web hook per la convalida delle firme.
- Kyverno: un motore di policy di Kubernetes configurato con un plug-in AWS Signer per la convalida delle firme.

> **Note**
>
> Prima di verificare le firme delle immagini del container, configura l'archivio di attendibilità e la policy di attendibilità di Notation, come richiesto dal controller di ammissione selezionato.

# Funzionalità EKS

> **Tip**
>
> Inizia: Crea funzionalità ACK | Crea funzionalitàArgo CD | Crea funzionalità kro

Amazon EKS Capabilities è un set stratificato di funzionalità di cluster completamente gestite che aiutano ad accelerare la velocità degli sviluppatori e a ridurre la complessità della creazione e della scalabilità con Kubernetes. EKS Capabilities sono Kubernetes-native funzionalità per la distribuzione continua dichiarativa, la gestione AWS delle risorse e la creazione e l'orchestrazione delle risorse Kubernetes, tutte completamente gestite da. AWS Con EKS Capabilities, puoi concentrarti maggiormente sulla creazione e sulla scalabilità dei tuoi carichi di lavoro, scaricando l'onere operativo di questi servizi di piattaforma fondamentali. AWS Queste funzionalità vengono eseguite all'interno di EKS anziché nei cluster, eliminando la necessità di installare, mantenere e scalare i componenti critici della piattaforma sui nodi di lavoro.

Per iniziare, puoi creare una o più funzionalità EKS su un cluster EKS nuovo o esistente. A tale scopo, puoi utilizzare la AWS CLI, le API EKS, eksctl o i Console di gestione AWS tuoi strumenti infrastructure-as-code preferiti. Sebbene EKS Capabilities sia progettato per funzionare insieme, sono risorse cloud indipendenti che puoi scegliere in base al tuo caso d'uso e ai tuoi requisiti.

Tutte le versioni di Kubernetes supportate da EKS sono supportate per EKS Capabilities.

> **Note**
>
> Le funzionalità EKS sono disponibili in tutte le regioni AWS commerciali in cui è disponibile Amazon EKS. Per un elenco delle regioni supportate, consulta gli endpoint e le quote di Amazon EKS nella Guida AWS generale.

## Funzionalità disponibili

### AWS Controller per Kubernetes (ACK)

ACK consente la gestione delle AWS risorse utilizzando le API Kubernetes, consentendoti di creare e gestire bucket S3, database RDS, ruoli IAM e altre risorse utilizzando risorse personalizzate

Kubernetes. AWS ACK riconcilia continuamente lo stato desiderato con lo stato effettivo in cui si trova AWS, correggendo eventuali variazioni nel tempo per mantenere il sistema integro e le risorse configurate come specificato. Puoi gestire AWS le risorse insieme ai carichi di lavoro Kubernetes utilizzando gli stessi strumenti e flussi di lavoro, con supporto per più di 50 AWS servizi tra cui S3, RDS, DynamoDB e Lambda. ACK supporta la gestione delle risorse tra account e regioni, abilitando complesse architetture di gestione di sistemi multi-account e multi-cluster. ACK supporta risorse di sola lettura e adozione in sola lettura, facilitando la migrazione da altre infrastrutture come strumenti di codice ai sistemi. Kubernetes-based

Scopri di più su ACK →

## Argo CD

Argo CD implementa la distribuzione GitOps-based continua per le tue applicazioni, utilizzando i repository Git come fonte di verità per i tuoi carichi di lavoro e lo stato del sistema. Argo CD sincronizza automaticamente le risorse delle applicazioni con i cluster dai repository Git, rilevando e correggendo le deviazioni per garantire che le applicazioni distribuite corrispondano allo stato desiderato. È possibile distribuire e gestire applicazioni su più cluster da una singola istanza Argo CD, con la distribuzione automatizzata dai repository Git ogni volta che vengono apportate modifiche. L'uso combinato di Argo CD e ACK fornisce un GitOps sistema fondamentale, che semplifica la gestione delle dipendenze dei carichi di lavoro e supporta progetti di sistemi completi, inclusa la gestione di cluster e infrastrutture su larga scala. Argo CD si integra con AWS Identity Center per l'autenticazione e l'autorizzazione e fornisce un'interfaccia utente Argo ospitata per visualizzare lo stato di integrità e di implementazione delle applicazioni.

Scopri di più su Argo CD →

## kro (Kube Resource Orchestrator)

kro ti consente di creare API Kubernetes personalizzate che compongono più risorse in astrazioni di livello superiore, permettendo ai team della piattaforma di definire modelli riutilizzabili per combinazioni di risorse comuni: elementi costitutivi del cloud. Con kro, puoi comporre sia Kubernetes che AWS le risorse insieme in astrazioni unificate, utilizzando una sintassi semplice per abilitare configurazioni dinamiche e logica condizionata. kro consente ai team della piattaforma di fornire funzionalità self-service con guardrail appropriati, consentendo agli sviluppatori di fornire infrastrutture complesse utilizzando API semplici e create appositamente mantenendo gli standard organizzativi e le migliori pratiche. le risorse kro sono semplicemente risorse Kubernetes e sono specificate in Kubernetes Mostra manifesti che possono essere archiviati in Git o inviati a OCI-compatible registri come Amazon ECR per un'ampia distribuzione organizzativa.

Scopri di più su kro →

# Vantaggi delle funzionalità EKS

Le funzionalità EKS sono completamente gestite da AWS, eliminando la necessità di installazione, manutenzione e scalabilità dei servizi cluster di base. AWS gestisce le patch di sicurezza, gli aggiornamenti e la gestione operativa, permettendo ai team di concentrarsi sulla creazione AWS anziché sulle operazioni del cluster. A differenza dei tradizionali componenti aggiuntivi di Kubernetes che consumano risorse del cluster, le funzionalità vengono eseguite in EKS anziché sui nodi di lavoro. Ciò consente di liberare capacità e risorse del cluster per i carichi di lavoro, riducendo al minimo l'onere operativo della gestione dei controller interni al cluster e di altri componenti della piattaforma.

Con EKS Capabilities, puoi gestire distribuzioni, AWS risorse Kubernetes personalizzate e composizioni utilizzando API e strumenti Kubernetes nativi come. `kubectl` Tutte le funzionalità operano nel contesto dei cluster, rilevando e correggendo automaticamente le variazioni di configurazione nelle risorse applicative e dell'infrastruttura cloud. È possibile distribuire e gestire le risorse su più cluster, AWS account e regioni da un unico punto di controllo, semplificando le operazioni in ambienti complessi e distribuiti.

Le funzionalità EKS sono progettate per i GitOps flussi di lavoro e forniscono infrastrutture dichiarative e con controllo della versione e una gestione delle applicazioni. Le modifiche fluiscono da Git attraverso il sistema, fornendo audit trail, funzionalità di rollback e flussi di lavoro di collaborazione che si integrano con le pratiche di sviluppo esistenti. Questo Kubernetes-native approccio significa che non è necessario utilizzare più strumenti o gestire sistemi infrastructure-as-code esterni ai cluster e che esiste un'unica fonte di verità a cui fare riferimento. Lo stato desiderato, definito nella configurazione dichiarativa di Kubernetes con controllo di versione, viene applicato continuamente in tutto l'ambiente.

# Prezzi

Con EKS Capabilities, non sono previsti impegni anticipati o commissioni minime. Ti viene addebitato un costo per ogni risorsa di capacità per ogni ora in cui è attiva sul tuo cluster Amazon EKS. Le risorse Kubernetes specifiche gestite da EKS Capabilities vengono inoltre fatturate su base oraria.

Per informazioni aggiornate sui prezzi, consulta la pagina dei prezzi di Amazon EKS.

> **Tip**
>
> È possibile utilizzare AWS Cost Explorer e Cost and Usage Reports per tenere traccia dei costi delle funzionalità separatamente dagli altri addebiti EKS. È possibile etichettare le funzionalità con il nome del cluster, il tipo di funzionalità e altri dettagli ai fini dell'allocazione dei costi.

## Come funzionano le funzionalità EKS

Ogni funzionalità è una AWS risorsa che crei sul tuo cluster EKS. Una volta creata, la funzionalità viene eseguita in EKS ed è completamente gestita da AWS.

> **Note**
>
> È possibile creare una risorsa di funzionalità di ogni tipo (Argo CD, ACK e kro) per un determinato cluster. Non è possibile creare risorse con funzionalità multiple dello stesso tipo sullo stesso cluster.

Interagisci con le funzionalità del tuo cluster utilizzando API e strumenti Kubernetes standard:

- Utilizzalo per applicare risorse personalizzate `kubectl` Kubernetes
- Usa gli archivi Git come fonte di verità per GitOps i flussi di lavoro

Alcune funzionalità supportano strumenti aggiuntivi. Esempio:

- Usa la CLI di Argo CD per configurare e gestire repository e cluster con le funzionalità di Argo CD
- Utilizzate l'interfaccia utente di Argo CD per visualizzare e gestire le applicazioni gestite dalla funzionalità Argo CD

Le funzionalità sono progettate per funzionare insieme, ma sono indipendenti e completamente attivabili. È possibile abilitare una, due o tutte e tre le funzionalità in base alle proprie esigenze e aggiornare la configurazione man mano che i requisiti evolvono.

Tutti i tipi di EKS Compute sono supportati per l'uso con EKS Capabilities. Per ulteriori informazioni, consulta Gestione del calcolo.

Per la configurazione di sicurezza e i dettagli sui ruoli IAM, consultathe section called "Considerazioni sulle funzionalità EKS". Per i modelli di architettura multi-cluster, vedithe section called "Considerazioni".

# Casi di utilizzo comune

GitOps per applicazioni e infrastruttura

Usa Argo CD per distribuire applicazioni e componenti operativi e ACK per gestire le configurazioni dei cluster e fornire l'infrastruttura, entrambi dai repository Git. L'intero stack (applicazioni, database, storage e rete) è definito come codice e distribuito automaticamente.

Esempio: un team di sviluppo invia modifiche a Git. Argo CD distribuisce l'applicazione aggiornata e ACK fornisce un nuovo database RDS con la configurazione corretta. Tutte le modifiche sono verificabili, reversibili e coerenti in tutti gli ambienti.

Progettazione della piattaforma con Self-Service

Usa kro per creare API personalizzate che compongono risorse ACK e Kubernetes. I team della piattaforma definiscono i modelli approvati con guardrail. I team applicativi utilizzano API semplici e di alto livello per fornire stack completi.

Esempio: un team della piattaforma crea un'API "WebApplication" che fornisce un bucket Deployment, Service, Ingress e S3. Gli sviluppatori utilizzano questa API senza dover comprendere la complessità o le autorizzazioni sottostanti. AWS

Multi-Cluster Gestione delle applicazioni

Usa Argo CD per distribuire applicazioni su più cluster EKS in diverse regioni o account. Gestisci tutte le implementazioni da una singola istanza Argo CD con policy e flussi di lavoro coerenti.

Esempio: distribuisci la stessa applicazione in cluster di sviluppo, gestione temporanea e produzione in più regioni. Argo CD assicura che ogni ambiente rimanga sincronizzato con il ramo Git corrispondente.

Multi-Cluster Gestione

Usa ACK per definire e fornire i cluster EKS, kro per personalizzare le configurazioni dei cluster con standard organizzativi e Argo CD per gestire il ciclo di vita e la configurazione dei cluster. Ciò fornisce una gestione completa dei cluster dalla creazione alle operazioni in corso.

Esempio: definizione dei cluster EKS utilizzando ACK e kro per fornire e gestire l'infrastruttura dei cluster, definire standard organizzativi per il networking, le politiche di sicurezza, i componenti aggiuntivi e altre configurazioni. Usa Argo CD per creare e gestire continuamente cluster, configurazioni e aggiornamenti delle versioni di Kubernetes in tutta la tua flotta sfruttando standard coerenti e la gestione automatizzata del ciclo di vita.

Migrazioni e modernizzazione

Semplifica la migrazione a EKS con il provisioning delle risorse cloud e i flussi di lavoro nativi. GitOps Usa ACK per adottare AWS le risorse esistenti senza ricrearle e Argo CD per rendere operative le distribuzioni dei carichi di lavoro da Git.

Esempio: un team che esegue la migrazione da EC2 a EKS adotta i database RDS e i bucket S3 esistenti utilizzando ACK, quindi utilizza Argo CD per distribuire applicazioni containerizzate da Git. Il percorso di migrazione è chiaro e le operazioni sono standardizzate sin dal primo giorno.

Account e Bootstrapping regionale

Automatizza l'implementazione dell'infrastruttura tra account e regioni utilizzando insieme Argo CD e ACK. Definisci la tua infrastruttura come codice in Git e lascia che siano le funzionalità a gestire l'implementazione e la gestione.

Esempio: un team della piattaforma gestisce repository Git che definiscono configurazioni di account standard: VPC, ruoli IAM, istanze RDS e stack di monitoraggio. Argo CD distribuisce automaticamente queste configurazioni su nuovi account e regioni, garantendo la coerenza e riducendo i tempi di configurazione manuale da giorni a minuti.

# Lavorare con risorse di capacità

Elenca, descrivi, aggiorna ed elimina le risorse di funzionalità sui tuoi cluster Amazon EKS.

## Risorse relative alle funzionalità EKS

Le funzionalità EKS sono AWS risorse che abilitano funzionalità gestite sul tuo cluster Amazon EKS. Le funzionalità vengono eseguite in EKS, eliminando la necessità di installare e mantenere controller e altri componenti operativi sui nodi di lavoro. Le funzionalità vengono create per un cluster EKS specifico e rimangono affiliate a quel cluster per l'intero ciclo di vita.

Ogni risorsa di funzionalità ha:

- Un nome univoco all'interno del cluster

- Un tipo di funzionalità (ACK, ARGOCD o KRO)
- Un Amazon Resource Name (ARN), che specifica sia il nome che il tipo
- Un ruolo IAM di capacità
- Uno stato che ne indica lo stato attuale
- Configurazione, sia generica che specifica per il tipo di funzionalità

## Comprensione dello stato della capacità

Lo stato delle risorse di capacità indica il loro stato attuale. È possibile visualizzare lo stato e l'integrità delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Funzionalità per visualizzare lo stato di tutte le funzionalità.
4. Per informazioni dettagliate sullo stato di salute, scegli la scheda Osservabilità, quindi il cluster di monitoraggio, quindi la scheda Capacità.

AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name
```

### Stati delle funzionalità

CREAZIONE: la capacità è in fase di configurazione. È possibile uscire dalla console: la funzionalità continuerà a creare in background.

ATTIVO: La funzionalità è attiva e pronta all'uso. Se le risorse non funzionano come previsto, controlla lo stato delle risorse e le autorizzazioni IAM. Per ulteriori informazioni, consulta the section called "Funzionalità di risoluzione dei problemi".

AGGIORNAMENTO: le modifiche alla configurazione sono in corso di applicazione. Attendi il ripristino dello stato`ACTIVE`.

ELIMINAZIONE: la funzionalità viene rimossa dal cluster.

CREATE_FAILED: il programma di installazione ha rilevato un errore. Le cause più comuni includono:

- Politica di fiducia dei ruoli IAM errata o mancante
- Il ruolo IAM non esiste o non è accessibile
- Problemi di accesso al cluster
- Parametri di configurazione non validi

Controlla la sezione sullo stato delle funzionalità per dettagli specifici sugli errori.

UPDATE_FAILED: aggiornamento della configurazione non riuscito. Controlla la sezione sullo stato delle funzionalità per i dettagli e verifica le autorizzazioni IAM.

> **Tip**
>
> Per una guida dettagliata alla risoluzione dei problemi, consulta:
>
> - the section called "Funzionalità di risoluzione dei problemi"- Risoluzione dei problemi relativi alle funzionalità generali
> - the section called "Risoluzione dei problemi"- problemi ACK-specific
> - the section called "Risoluzione dei problemi"- Problemi con Argo CD-specific
> - the section called "Risoluzione dei problemi"- problemi kro specifici

## Crea funzionalità

Per creare una funzionalità sul tuo cluster, consulta i seguenti argomenti:

- the section called "Crea funzionalità ACK"— Creare una funzionalità ACK per gestire AWS le risorse utilizzando le API Kubernetes
- the section called "Crea una funzionalità Argo CD"— Creare una funzionalità Argo CD per la distribuzione continua GitOps
- the section called "Crea una capacità kro"— Creare una funzionalità kro per la composizione e l'orchestrazione delle risorse

# Elenca le funzionalità

È possibile elencare tutte le risorse di funzionalità in un cluster.

## Console

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Visualizza le risorse relative alle funzionalità in Funzionalità gestite.

## AWS CLI

Usa il `list-capabilities` comando per visualizzare tutte le funzionalità del tuo cluster. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscilo *`my-cluster`* con il nome del cluster.

```
aws eks list-capabilities \
  --region region-code \
  --cluster-name my-cluster
```

```
{
    "capabilities": [
        {
            "capabilityName": "my-ack",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/ack/my-
ack/abc123",
            "type": "ACK",
            "status": "ACTIVE",
            "createdAt": "2025-11-02T10:30:00.000000-07:00",
            "modifiedAt": "2025-11-02T10:32:15.000000-07:00"
        },
        {
            "capabilityName": "my-kro",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/kro/my-
kro/abc123",
            "type": "KRO",
            "status": "ACTIVE",
            "version": "v0.6.3",
```

```
            "createdAt": "2025-11-02T10:30:00.000000-07:00",
            "modifiedAt": "2025-11-02T10:32:15.000000-07:00"
        },
        {
            "capabilityName": "my-argocd",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/argocd/my-
argocd/abc123",
            "type": "ARGOCD",
            "status": "ACTIVE",
            "version": "3.1.8-eks-1",
            "createdAt": "2025-11-21T08:22:28.486000-05:00",
            "modifiedAt": "2025-11-21T08:22:28.486000-05:00"
        }
    ]
}
```

## Descrivi una funzionalità

Ottieni informazioni dettagliate su una funzionalità specifica, inclusa la sua configurazione e lo stato.

### Console

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la  scheda  Funzionalità.
4. Scegli la funzionalità che desideri visualizzare tra le funzionalità  gestite.
5. Visualizza i dettagli delle funzionalità, tra cui stato, configurazione e ora di creazione.

### AWS CLI

Usa il `describe-capability` comando per visualizzare informazioni dettagliate. Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster, sostituisci *`my-cluster`* con il nome del tuo cluster e sostituisci *`capability-name`* con il nome che hai dato alla tua capacità quando l'hai creata (ad esempio`my-ack`,`my-argocd`, o`my-kro`).

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

Output di esempio:


```
{
  "capability": {
    "capabilityName": "my-ack",
    "capabilityArn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/ack/my-
ack/abc123",
    "clusterName": "my-cluster",
    "type": "ACK",
    "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSCapabilityACKRole",
    "status": "ACTIVE",
    "configuration": {},
    "tags": {},
    "health": {
      "issues": []
    },
    "createdAt": "2025-11-19T17:11:30.242000-05:00",
    "modifiedAt": "2025-11-19T17:11:30.242000-05:00",
    "deletePropagationPolicy": "RETAIN"
  }
}
```


## Aggiorna la configurazione di una funzionalità

È possibile aggiornare alcuni aspetti della configurazione di una funzionalità dopo la creazione. Le opzioni di configurazione specifiche variano in base al tipo di funzionalità.

> **Note**
>
> Le risorse di funzionalità EKS sono completamente gestite, inclusi l'applicazione di patch e gli aggiornamenti delle versioni. L'aggiornamento di una funzionalità aggiornerà la configurazione delle risorse e non comporterà aggiornamenti di versione dei componenti delle funzionalità gestite.

### AWS CLI

Usa il `update-capability` comando per modificare una funzionalità:

```
aws eks update-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name \
  --role-arn arn:aws:iam::111122223333:role/NewCapabilityRole
```

> **Note**
>
> Non tutte le proprietà delle funzionalità possono essere aggiornate dopo la creazione. Per informazioni dettagliate su cosa può essere modificato, consulta la documentazione specifica della funzionalità.

## Eliminare una funzionalità

Quando non è più necessaria una funzionalità sul cluster, è possibile eliminare la risorsa di funzionalità.

> **Important**
>
> Elimina le risorse del cluster prima di eliminare la funzionalità.
> L'eliminazione di una risorsa di capacità non comporta automaticamente l'eliminazione delle risorse create tramite tale capacità:
>
> - Tutte le Kubernetes Custom Resource Definitions (CRD) rimangono installate nel cluster.
> - Le risorse ACK rimangono nel cluster e le AWS risorse corrispondenti rimangono nel tuo account
> - Le applicazioni Argo CD e le relative risorse Kubernetes rimangono nel cluster
> - kro ResourceGraphDefinitions e le istanze rimangono nel tuo cluster
>
> È necessario eliminare queste risorse prima di eliminare la funzionalità per evitare risorse orfane.
> Facoltativamente, puoi scegliere di conservare AWS le risorse associate alle risorse ACK Kubernetes. Vedi le considerazioni su ACK ???

### Console

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Seleziona la funzionalità che desideri eliminare dall'elenco delle funzionalità gestite.
5. Scegli la funzionalità Elimina.
6. Nella finestra di dialogo di conferma, digita il nome della funzionalità per confermare l'eliminazione.
7. Scegliere Elimina.

### AWS CLI

Usa il `delete-capability` comando per eliminare una risorsa di funzionalità:

Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster, sostituisci *`my-cluster`* con il nome del tuo cluster e sostituisci *`capability-name`* con il nome della capacità da eliminare.

```
aws eks delete-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

## Fasi successive

- the section called "risorse Kubernetes"— Scopri le risorse Kubernetes fornite da ciascun tipo di funzionalità
- the section called "Concetti ACK"— Comprendi i concetti di ACK e il ciclo di vita delle risorse
- the section called "Lavorare con Argo CD"— Utilizzo delle funzionalità di Argo CD per i flussi di lavoro GitOps
- the section called "concetti kro"— Comprendi i concetti di kro e la composizione delle risorse

# Accedi ai registri del controller EKS Capabilities

I controller EKS Capabilities per ACK, kro e Argo CD vengono eseguiti in un'infrastruttura gestita all'esterno dei cluster. AWS Puoi configurare la consegna dei log per questi controller utilizzando

Amazon CloudWatch Vending Logs, che ti offre visibilità sul comportamento dei controller per il monitoraggio e la risoluzione dei problemi.

I log vengono forniti in formato JSON strutturato e includono campi operativi come livello di log, messaggio, nome del controller e identificatori di riconciliazione. AWS I metadati interni vengono filtrati prima della consegna: l'utente riceve solo il contenuto dei log operativi pertinenti ai controller di capacità.

## Tipi di log supportati

Ogni funzionalità ha uno o più tipi di log che è possibile configurare indipendentemente come fonti di distribuzione di CloudWatch Vend Logs.

ACK

- EKS_CAPABILITY_ACK_LOGS

kro

- EKS_CAPABILITY_KRO_LOGS

CD Argo

- REGISTRI DELLE APPLICAZIONI EKS_CAPABILITY_ARGOCD
- EKS_CAPABILITY_ARGOCD_APPLICATIONSET_LOGS
- EKS_CAPABILITY_ARGOCD_COMMITSERVER_LOGS
- EKS_CAPABILITY_ARGOCD_REPOSERVER_LOGS
- EKS_CAPABILITY_ARGOCD_SERVER_LOGS

ACK utilizza un unico tipo di registro che copre tutti i controller di servizio ACK. I record di registro includono un `controllerGroup` campo (ad esempio`s3.services.k8s.aws`,`rds.services.k8s.aws`) che identifica quale controller di servizio ACK ha generato il registro. È possibile utilizzare questo campo per filtrare i log per un controller di servizio specifico in CloudWatch Logs Insights o in altri strumenti di interrogazione.

Argo CD ha cinque tipi di log separati, uno per componente del controller. Ciò consente di abilitare la registrazione solo per i controller necessari e di indirizzarli verso destinazioni diverse.

# Impostazione della consegna dei log

Per configurare la distribuzione dei log del controller di capacità, usa Console di gestione AWS o l'API Amazon CloudWatch Logs. Per istruzioni dettagliate sulla configurazione, consulta Enabling logging from AWS services nella Amazon CloudWatch Logs User Guide. Ogni controller di funzionalità può essere configurato come una singola fonte di distribuzione di CloudWatch Vend Logs, in modo da poter selezionare quali log ricevere.

## Console

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Funzionalità, quindi seleziona la tua capacità.
4. Nella sezione Consegna del registro, scegli Aggiungi.
5. Seleziona il tipo di registro adatto alla tua capacità e scegli una destinazione.
6. Scegli Aggiungi per creare la consegna.

## Utilizzo delle CloudWatch API Amazon

La configurazione della registrazione richiede tre passaggi:

1. Crea una fonte di distribuzione per la funzionalità utilizzando l' CloudWatch `PutDeliverySource`API. Usa la funzionalità ARN come `resourceArn` e specifica il tipo di registro desiderato.
2. Crea una destinazione di consegna utilizzando`PutDeliveryDestination`. Specificare l'ARN di un gruppo di CloudWatch log, di un bucket S3 o di un flusso di distribuzione di Amazon Data Firehose.
3. Crea una consegna per connettere l'origine e la destinazione utilizzando. `CreateDelivery`

È possibile recuperare la capacità ARN utilizzando `describe-capability` il comando:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability \
  --query 'capability.capabilityArn' --output text
```

È possibile configurare una singola funzionalità (origine di consegna) per inviare registri a più destinazioni creando più consegne. Puoi anche inviare registri da più funzionalità alla stessa destinazione.

### autorizzazioni IAM

A seconda della destinazione selezionata, potrebbe essere necessario configurare le policy o i ruoli IAM per il gruppo di CloudWatch log, il bucket S3 o il flusso di distribuzione Data Firehose per garantire una corretta consegna dei log. Se invii log tra AWS account, utilizza l'`PutDeliveryDestinationPolicy`API per configurare una policy IAM che consenta la consegna alla destinazione. Per ulteriori informazioni, consulta la documentazione sulle autorizzazioni di CloudWatch Vending Logs.

## Esempio: interrogazione dei log con Logs Insights CloudWatch

Per visualizzare i log degli errori su tutti i controller:

```
fields @timestamp, controller, message, error
| filter level = "error"
| sort @timestamp desc
| limit 50
```

Per altri esempi di query, tra cui il filtraggio per controller di servizio, il monitoraggio dei cicli di riconciliazione e il filtraggio delle applicazioni Argo CD, consulta Utilizzare i registri dei controller per la risoluzione dei problemi.

## Visualizzazione dei log

Dopo aver configurato la consegna dei log, i log vengono recapitati alla destinazione specificata. Il metodo di accesso ai log dipende dal tipo di destinazione scelto:

- CloudWatch Registri: visualizza i log nella console Logs, usa CloudWatch i comandi AWS CLI o esegui query con Logs Insights. CloudWatch
- Amazon S3: accedi ai log come oggetti nel tuo bucket S3 tramite la console S3, l'interfaccia a riga di AWS comando o strumenti di analisi come Amazon Athena.
- Amazon Data Firehose: i log vengono trasmessi in streaming al target Firehose configurato (ad esempio S3, Service OpenSearch , Redshift, ecc.).

## Prezzi

CloudWatch Le tariffe di Vending Logs si applicano per la consegna e lo stoccaggio dei registri in base alla destinazione di consegna prescelta. CloudWatch Vending Logs consente una consegna affidabile e sicura dei log con AWS autenticazione e autorizzazione integrate a un prezzo ridotto rispetto ai registri standard. CloudWatch Per maggiori dettagli, consulta la sezione Vending Logs della pagina dei CloudWatch prezzi.

# Capability: risorse Kubernetes

Dopo aver abilitato una funzionalità sul cluster, molto spesso interagirai con essa creando e gestendo risorse personalizzate Kubernetes nel cluster. Ogni funzionalità fornisce il proprio set di definizioni di risorse personalizzate (CRD) che estendono l'API Kubernetes con funzionalità specifiche per funzionalità.

## Risorse Argo CD

Quando abiliti la funzionalità Argo CD, puoi creare e gestire le seguenti risorse Kubernetes:

Applicazione

Definisce una distribuzione da un repository Git a un cluster di destinazione. `Application`le risorse specificano il repository di origine, lo spazio dei nomi di destinazione e la politica di sincronizzazione. È possibile creare fino a 3.000 `Application` risorse per istanza con funzionalità Argo CD.

ApplicationSet

Genera più `Application` risorse a partire da modelli, abilitando implementazioni multi-cluster e multiambiente. `ApplicationSet`le risorse utilizzano i generatori per creare `Application` risorse in modo dinamico sulla base di elenchi di cluster, directory Git o altre fonti.

AppProject

Fornisce il raggruppamento logico e il controllo degli accessi alle risorse. `Application` `AppProject`le risorse definiscono quali repository, cluster e namespace possono essere utilizzati dalle `Application` risorse, abilitando limiti di multi-tenancy e sicurezza.

`Application`Risorsa di esempio:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/org/repo
    targetRevision: main
    path: manifests
  destination:
    server: https://kubernetes.default.svc
    namespace: production
```

Per ulteriori informazioni sulle risorse e sui concetti di Argo CD, vederethe section called "Concetti di Argo CD".

## risorse kro

Quando abiliti la funzionalità kro, puoi creare e gestire le seguenti risorse Kubernetes:

ResourceGraphDefinition (RGD)

Definisce un'API personalizzata che compone più Kubernetes e AWS risorse in un'astrazione di livello superiore. I team della piattaforma creano `ResourceGraphDefinition` risorse per fornire modelli riutilizzabili con guardrail.

Istanze di risorse personalizzate

Dopo aver creato una `ResourceGraphDefinition` risorsa, puoi creare istanze dell'API personalizzata definita da`ResourceGraphDefinition`. kro crea e gestisce automaticamente le risorse specificate in. `ResourceGraphDefinition`

Risorsa di esempio`ResourceGraphDefinition`:

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: web-application
spec:
```

```
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    spec:
      name: string
      replicas: integer
  resources:
    - id: deployment
      template:
        apiVersion: apps/v1
        kind: Deployment
        # ... deployment spec
    - id: service
      template:
        apiVersion: v1
        kind: Service
        # ... service spec
```

Istanza di `WebApplication` esempio:

```
apiVersion: v1alpha1
kind: WebApplication
metadata:
  name: my-web-app
  namespace: default
spec:
  name: my-web-app
  replicas: 3
```

Quando applichi questa istanza, kro crea automaticamente `Service` le risorse `Deployment` e definite in. `ResourceGraphDefinition`

Per ulteriori informazioni sulle risorse e sui concetti di kro, consulta. the section called "concetti kro"

## Risorse ACK

Quando abiliti la funzionalità ACK, puoi creare e gestire AWS risorse utilizzando risorse personalizzate Kubernetes. ACK fornisce oltre 200 CRD per più di 50 AWS servizi, consentendoti di definire AWS le risorse insieme ai carichi di lavoro Kubernetes e di gestire le risorse di infrastruttura dedicate con Kubernetes. AWS

Esempi di risorse ACK:

Bucket S3

`Bucket`le risorse creano e gestiscono bucket Amazon S3 con politiche di controllo delle versioni, crittografia e ciclo di vita.

Istanza DB RDS

`DBInstance`fornitura di risorse e gestione di istanze di database Amazon RDS con backup automatici e finestre di manutenzione.

Tabella DynamoDB

`Table`le risorse creano e gestiscono tabelle DynamoDB con capacità fornita o su richiesta.

Ruolo IAM

`Role`le risorse definiscono i ruoli IAM con policy di fiducia e policy di autorizzazione per l'accesso ai servizi. AWS

Funzione Lambda

`Function`le risorse creano e gestiscono funzioni Lambda con configurazione di codice, runtime e ruolo di esecuzione.

Esempio di specificazione di una `Bucket` risorsa:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-app-bucket
spec:
  name: my-unique-bucket-name-12345
  versioning:
    status: Enabled
  encryption:
    rules:
      - applyServerSideEncryptionByDefault:
          sseAlgorithm: AES256
```

Per ulteriori informazioni sulle risorse e sui concetti di ACK, vederethe section called "Concetti ACK".

## Limiti delle risorse

Le funzionalità EKS hanno i seguenti limiti di risorse:

Limiti di utilizzo di Argo CD:

- Massimo 3.000 `Application` risorse per istanza con funzionalità Argo CD
- Massimo 100 cluster remoti configurati per istanza con funzionalità Argo CD

Limiti di configurazione delle risorse:

- Massimo 150 risorse Kubernetes per `Application` risorsa in Argo CD
- Massimo 64 risorse Kubernetes per mese `ResourceGraphDefinition`

> **Note**
>
> Questi limiti si applicano al numero di risorse gestite da ciascuna istanza di funzionalità. Se hai bisogno di limiti più elevati, puoi distribuire funzionalità su più cluster.

### Fasi successive

Per le attività specifiche relative alle funzionalità e la configurazione avanzata, consulta i seguenti argomenti:

- the section called "Concetti ACK"— Comprendi i concetti e il ciclo di vita delle risorse di ACK
- the section called "Lavorare con Argo CD"— Utilizzo delle funzionalità di Argo CD per i flussi di lavoro GitOps
- the section called "concetti kro"— Comprendi i concetti kro e la composizione delle risorse

## Considerazioni sulle funzionalità EKS

Esamina le considerazioni importanti prima di implementare le funzionalità EKS, tra cui la progettazione del controllo degli accessi, la scelta tra soluzioni gestite e autogestite, i modelli architettonici per le implementazioni multi-cluster e le migliori pratiche operative.

### Capacità, ruoli IAM e Kubernetes RBAC

Ogni risorsa di funzionalità EKS ha un ruolo IAM con funzionalità configurate. Il ruolo di capacità viene utilizzato per concedere le autorizzazioni di AWS servizio alle funzionalità EKS di agire per

conto dell'utente. Ad esempio, per utilizzare EKS Capability for ACK per gestire i bucket Amazon S3, concederai le autorizzazioni amministrative di S3 Bucket alla funzionalità, consentendole di creare e gestire i bucket.

Una volta configurata la funzionalità, le risorse S3 AWS possono essere create e gestite con le risorse personalizzate Kubernetes presenti nel cluster. Kubernetes RBAC è il meccanismo di controllo degli accessi all'interno del cluster che consente di determinare quali utenti e gruppi possono creare e gestire tali risorse personalizzate. Ad esempio, concedi a utenti e gruppi Kubernetes RBAC specifici le autorizzazioni per creare e gestire risorse nei namespace di tua scelta. `Bucket`

In questo modo, IAM e Kubernetes RBAC sono due parti del sistema di controllo degli accessi end-to-end che regola le autorizzazioni relative alle funzionalità e alle risorse EKS. È importante progettare la giusta combinazione di autorizzazioni IAM e policy di accesso RBAC per il tuo caso d'uso.

Per ulteriori informazioni sulla capacità, i ruoli IAM e le autorizzazioni Kubernetes, consulta. the section called "Considerazioni sulle funzionalità EKS"

## Multi-cluster modelli di architettura

Quando implementate funzionalità su più cluster, prendete in considerazione questi modelli architettonici comuni:

Hub and Spoke con gestione centralizzata

Esegui tutte e tre le funzionalità in un cluster gestito centralmente per orchestrare i carichi di lavoro e gestire l'infrastruttura cloud su più cluster di carichi di lavoro.

- Argo CD sul cluster di gestione distribuisce applicazioni su cluster di carico di lavoro in diverse regioni o account
- ACK sul cluster di gestione fornisce AWS risorse (RDS, S3, IAM) per tutti i cluster
- kro sul cluster di gestione crea astrazioni di piattaforma portatili che funzionano su tutti i cluster

Questo modello centralizza il carico di lavoro e la gestione dell'infrastruttura cloud e può semplificare le operazioni per le organizzazioni che gestiscono molti cluster.

Decentralizzato GitOps

I carichi di lavoro e l'infrastruttura cloud sono gestiti dalle funzionalità dello stesso cluster in cui vengono eseguiti i carichi di lavoro.

- Argo CD gestisce le risorse delle applicazioni sul cluster locale.
- Le risorse ACK vengono utilizzate per esigenze di cluster e carichi di lavoro.
- Le astrazioni della piattaforma kro vengono installate e orchestrano le risorse locali.

Questo modello decentralizza le operazioni, con i team che gestiscono i propri servizi di piattaforma dedicati in uno o più cluster.

Hub and Spoke con implementazione ACK ibrida

Combina modelli centralizzati e decentralizzati, con implementazioni di applicazioni centralizzate e gestione delle risorse in base all'ambito e alla proprietà.

- Cluster di hub:
  - Argo CD gestisce le GitOps implementazioni nel cluster locale e in tutti i cluster di carichi di lavoro remoti
  - ACK viene utilizzato nel cluster di gestione per risorse con ambito amministrativo (database di produzione, ruoli IAM, VPC)
  - kro viene utilizzato nel cluster di gestione per le astrazioni riutilizzabili della piattaforma
- Cluster Spoke:
  - I carichi di lavoro sono gestiti tramite Argo CD sul cluster di hub centralizzato
  - ACK viene utilizzato localmente per le risorse relative ai carichi di lavoro (bucket S3, istanze, code SQS) ElastiCache
  - kro viene utilizzato localmente per la composizione delle risorse e i modelli di blocchi costitutivi

Questo modello separa le preoccupazioni: i team di piattaforma gestiscono l'infrastruttura critica centralmente sui cluster di gestione, includendo facoltativamente i cluster di carichi di lavoro, mentre i team applicativi specificano e gestiscono le risorse cloud insieme ai carichi di lavoro.

Scelta di un pattern

Considerate questi fattori nella scelta di un'architettura:

- Struttura organizzativa: i team di piattaforma centralizzati preferiscono modelli di hub; i team decentralizzati possono preferire funzionalità per cluster

- Ambito delle Admin-scoped risorse: le risorse (database, IAM) spesso traggono vantaggio dalla gestione centralizzata; le risorse del carico di lavoro (bucket, code) possono essere gestite localmente
- Self-service: I team di piattaforma centralizzata possono creare e distribuire risorse prescrittive personalizzate per consentire il self-service sicuro delle risorse cloud per le esigenze di carichi di lavoro comuni
- Gestione della flotta di cluster: i cluster di gestione centralizzati forniscono un piano di controllo di proprietà del cliente per la gestione della flotta di cluster EKS, insieme ad altre risorse con ambito amministrativo
- Requisiti di conformità: alcune organizzazioni richiedono un controllo centralizzato per l'audit e la governance
- Complessità operativa: un minor numero di istanze di funzionalità semplifica le operazioni ma può creare colli di bottiglia

> **Note**
>
> Puoi iniziare con uno schema ed evolvere verso un altro man mano che la tua piattaforma matura. Le funzionalità sono indipendenti: puoi distribuirle in modo diverso tra i cluster in base alle tue esigenze.

## Confronto tra le funzionalità EKS e le soluzioni autogestite

EKS Capabilities offre esperienze completamente gestite per i più diffusi strumenti e controller Kubernetes eseguiti in EKS. Ciò si differenzia dalle soluzioni autogestite, che vengono installate e utilizzate nel cluster.

### Differenze principali

Implementazione e gestione

AWS gestisce completamente EKS Capabilities senza che sia richiesta l'installazione, la configurazione o la manutenzione dei componenti software. AWS installa e gestisce automaticamente tutte le Kubernetes Custom Resource Definitions (CRD) richieste nel cluster.

Con le soluzioni autogestite, puoi installare e configurare il software del cluster utilizzando Helm charts, kubectl o altri operatori. Hai il pieno controllo sul ciclo di vita del software e sulla

configurazione di runtime delle tue soluzioni autogestite, fornendo personalizzazioni a qualsiasi livello della soluzione.

Operazioni e manutenzione

AWS gestisce l'applicazione di patch e altre operazioni del ciclo di vita del software per EKS Capabilities, con aggiornamenti automatici e patch di sicurezza. Le funzionalità EKS sono integrate con AWS funzionalità per configurazioni semplificate, forniscono elevata disponibilità e tolleranza ai guasti integrate ed eliminano la risoluzione dei problemi all'interno del cluster dei carichi di lavoro dei controller.

Self-managed le soluzioni richiedono il monitoraggio dello stato e dei registri dei componenti, l'applicazione di patch di sicurezza e aggiornamenti delle versioni, la configurazione dell'alta disponibilità con più repliche e budget relativi alle interruzioni dei pod, la risoluzione dei problemi e la risoluzione dei problemi relativi al carico di lavoro dei controller e la gestione di release e versioni. Avete il pieno controllo sulle vostre implementazioni, ma ciò richiede spesso soluzioni su misura per l'accesso privato ai cluster e altre integrazioni che devono essere in linea con gli standard organizzativi e i requisiti di conformità di sicurezza.

Consumo di risorse

EKS Capabilities viene eseguito in EKS e all'esterno dei cluster, liberando risorse dei nodi e delle risorse del cluster. Le funzionalità non utilizzano risorse del carico di lavoro del cluster, non consumano CPU o memoria sui nodi di lavoro, scalano automaticamente e hanno un impatto minimo sulla pianificazione della capacità del cluster.

Self-managed le soluzioni eseguono controller e altri componenti sui nodi di lavoro, consumando direttamente le risorse dei nodi di lavoro, gli IP del cluster e altre risorse del cluster. La gestione dei servizi cluster richiede la pianificazione della capacità per i relativi carichi di lavoro e richiede la pianificazione e la configurazione delle richieste e dei limiti delle risorse per gestire i requisiti di scalabilità e alta disponibilità.

Supporto funzionalità

Essendo funzionalità di servizio completamente gestite, EKS Capabilities sono per loro natura convincenti rispetto alle soluzioni autogestite. Sebbene le funzionalità supportino la maggior parte delle funzionalità e dei casi d'uso, vi sarà una differenza nella copertura rispetto alle soluzioni autogestite.

Con le soluzioni autogestite, puoi controllare completamente la configurazione, le funzionalità opzionali e altri aspetti della funzionalità del tuo software. È possibile scegliere di eseguire immagini

personalizzate, personalizzare tutti gli aspetti della configurazione e controllare completamente la funzionalità della soluzione autogestita.

Considerazioni sui costi

Ogni risorsa di funzionalità EKS ha un costo orario correlato, che varia in base al tipo di funzionalità. Le risorse del cluster gestite dalla funzionalità hanno inoltre un costo orario associato ai rispettivi prezzi. Per ulteriori informazioni, consultare Prezzi di Amazon EKS.

Self-managed le soluzioni non hanno costi diretti legati agli AWS addebiti, ma pagano per le risorse di elaborazione del cluster utilizzate dai controller e dai relativi carichi di lavoro. Oltre al consumo di risorse di nodi e cluster, il costo totale di proprietà delle soluzioni autogestite include i costi operativi e le spese di manutenzione, risoluzione dei problemi e supporto.

## Scelta tra EKS Capabilities e soluzioni autogestite

EKS Capabilities Prendete in considerazione questa scelta quando desiderate ridurre il sovraccarico operativo e concentrarvi sul valore differenziato del software e dei sistemi, piuttosto che sulle operazioni della piattaforma in cluster per requisiti fondamentali. Utilizzate EKS Capabilities quando desiderate ridurre al minimo l'onere operativo delle patch di sicurezza e della gestione del ciclo di vita del software, liberare risorse di nodi e cluster per i carichi di lavoro delle applicazioni, semplificare la configurazione e la gestione della sicurezza e beneficiare della copertura di supporto. AWS Le funzionalità EKS sono ideali per la maggior parte dei casi d'uso in produzione e rappresentano l'approccio consigliato per le nuove implementazioni.

Self-managed soluzioni Prendi in considerazione questa scelta quando hai bisogno di versioni specifiche dell'API di risorsa Kubernetes, build di controller personalizzate, se disponi di strumenti e automazione esistenti basati su implementazioni autogestite o hai bisogno di una personalizzazione approfondita delle configurazioni di runtime dei controller. Self-managed le soluzioni offrono flessibilità per casi d'uso specializzati e hai il controllo completo sulla configurazione di implementazione e runtime.

> **Note**
>
> EKS Capabilities può coesistere nel cluster con soluzioni autogestite ed è possibile effettuare migrazioni graduali.

### Capability-specific confronti

Per confronti dettagliati, tra cui funzionalità specifiche relative alle funzionalità, differenze a monte e percorsi di migrazione, consulta:

- the section called "Confronto con l'autogestito"
- the section called "Confronto con l'autogestito"
- the section called "Confronto con quello autogestito"

# Implementazione AWS risorse da Kubernetes con AWS Controller per Kubernetes (ACK)

AWS Controllers for Kubernetes (ACK) consente di definire e gestire le risorse di servizio direttamente da Kubernetes. AWS Con AWS Controllers for Kubernetes (ACK), puoi gestire le risorse dei carichi di lavoro e l'infrastruttura cloud utilizzando le risorse personalizzate di Kubernetes, insieme ai carichi di lavoro delle tue applicazioni utilizzando le API e gli strumenti Kubernetes familiari.

Con EKS Capabilities, ACK è completamente gestito da, eliminando la necessità di installare AWS, mantenere e scalare i controller ACK sui cluster.

## Come funziona ACK

ACK traduce le specifiche delle risorse personalizzate di Kubernetes in chiamate API. AWS Quando crei, aggiorni o elimini una risorsa personalizzata Kubernetes che rappresenta una risorsa di AWS servizio, ACK effettua le chiamate AWS API richieste per creare, aggiornare o eliminare la risorsa. AWS

Ogni AWS risorsa supportata da ACK ha una propria definizione di risorsa personalizzata (CRD) che definisce lo schema dell'API Kubernetes per specificarne la configurazione. Ad esempio, ACK fornisce CRD per S3, inclusi bucket, bucket policy e altre risorse S3.

ACK riconcilia continuamente lo stato delle AWS risorse con lo stato desiderato definito nelle risorse personalizzate di Kubernetes. Se una risorsa esce dallo stato desiderato, ACK lo rileva e intraprende azioni correttive per riportarla all'allineamento. Le modifiche alle risorse Kubernetes si riflettono immediatamente sullo stato AWS delle risorse, mentre il rilevamento passivo della deriva e la correzione delle modifiche alle AWS risorse a monte possono richiedere fino a 10 ore (il periodo di risincronizzazione), ma in genere avvengono molto prima.

Esempio di manifesto delle risorse S3 Bucket

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-ack-bucket
spec:
  name: my-unique-bucket-name
```

Quando applichi questa risorsa personalizzata al tuo cluster, ACK crea un bucket Amazon S3 nel tuo account se non esiste ancora. Le modifiche successive a questa risorsa, ad esempio specificando un livello di storage non predefinito o aggiungendo una policy, verranno applicate alla risorsa S3 in. AWS Quando questa risorsa viene eliminata dal cluster, il bucket in AWS S3 viene eliminato per impostazione predefinita.

## Vantaggi dell'ACK

ACK fornisce la gestione Kubernetes-native AWS delle risorse, consentendoti di gestire AWS le risorse utilizzando le stesse API e gli stessi strumenti Kubernetes che utilizzi per le tue applicazioni. Questo approccio unificato semplifica il flusso di lavoro di gestione dell'infrastruttura eliminando la necessità di passare da uno strumento all'altro o di apprendere sistemi infrastructure-as-code separati. Definisci AWS le tue risorse in modo dichiarativo nei manifest di Kubernetes, abilitando i GitOps flussi di lavoro e l'infrastruttura come pratiche di codice che si integrano perfettamente con i processi di sviluppo esistenti.

ACK riconcilia continuamente lo stato desiderato delle AWS risorse con il loro stato effettivo, correggendo la deriva e garantendo la coerenza in tutta l'infrastruttura. Questa riconciliazione continua significa che le modifiche imperative fuori banda alle AWS risorse vengono automaticamente ripristinate in modo che corrispondano alla configurazione dichiarata, mantenendo l'integrità dell'infrastruttura come codice. È possibile configurare ACK per gestire le risorse su più AWS account e regioni, abilitando complesse architetture multi-account senza strumenti aggiuntivi.

Per le organizzazioni che stanno migrando da altri strumenti di gestione dell'infrastruttura, ACK supporta l'adozione delle risorse, consentendo di riportare AWS le risorse esistenti sotto la gestione ACK senza ricrearle nuovamente. ACK fornisce anche risorse di sola lettura per l'osservazione AWS delle risorse senza accesso alle modifiche e annotazioni per conservare facoltativamente le AWS risorse anche quando la risorsa Kubernetes viene eliminata dal cluster.

Per saperne di più e iniziare a usare EKS Capability for ACK, vedi e. the section called "Concetti ACK" the section called "Considerazioni"

## Supportata AWS services

ACK supporta un'ampia gamma di AWS servizi, tra cui, a titolo esemplificativo ma non esaustivo:

- Amazon EC2
- Simple Storage Service (Amazon S3)
- Amazon RDS
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon EKS
- Amazon SQS
- Amazon SNS
- AWS Lambda
- AWS IO SONO

Tutti i AWS servizi elencati come Generally Available upstream sono supportati da EKS Capability for ACK. Per l'elenco completo dei AWS servizi supportati, consulta la pagina dei servizi ACK sul sito Web ACK.

## Integrazione con altre funzionalità gestite da EKS

ACK si integra con altre funzionalità gestite da EKS.

- Argo CD: utilizza Argo CD per gestire l'implementazione delle risorse ACK su più cluster, abilitando GitOps i flussi di lavoro per la tua infrastruttura. AWS
  - ACK estende i vantaggi dell' GitOps abbinamento con Argo CD, ma ACK non richiede l'integrazione con git.
- kro (Kube Resource Orchestrator): usa kro per comporre risorse complesse a partire da risorse ACK, creando astrazioni di livello superiore che semplificano la gestione delle risorse.
  - Puoi creare risorse personalizzate composite con kro che definiscono sia le risorse che le risorse Kubernetes. AWS I membri del team possono utilizzare queste risorse personalizzate per implementare rapidamente applicazioni complesse.

# Guida introduttiva a ACK

Per iniziare a usare EKS Capability for ACK:

1. Crea e configura un IAM Capability Role con le autorizzazioni necessarie affinché ACK possa gestire AWS le risorse per tuo conto.
2. Crea una risorsa di funzionalità ACK sul tuo cluster EKS tramite la AWS Console, la AWS CLI o la tua infrastruttura preferita come strumento di codice.
3. Applica le risorse personalizzate di Kubernetes al tuo cluster per iniziare a gestire le tue risorse in Kubernetes. AWS

# Crea una funzionalità ACK

Questo capitolo spiega come creare una funzionalità ACK sul tuo cluster Amazon EKS.

## Prerequisiti

Prima di creare una funzionalità ACK, assicurati di avere:

- Cluster Amazon EKS
- Un ruolo di capacità IAM con autorizzazioni per ACK per gestire le AWS risorse
- Autorizzazioni IAM sufficienti per creare risorse di funzionalità sui cluster EKS
- Lo strumento CLI appropriato installato e configurato o l'accesso alla console EKS

Per istruzioni sulla creazione dello IAM Capability Role, consultathe section called "Capacità, ruolo IAM".

> **Important**
>
> ACK è una funzionalità di gestione dell'infrastruttura che consente di creare, modificare ed eliminare AWS risorse. Si tratta di una funzionalità amministrativa che deve essere controllata attentamente. Chiunque sia autorizzato a creare risorse Kubernetes nel tuo cluster può creare risorse in modo efficace tramite ACK, fatte salve AWS le autorizzazioni IAM Capability Role. Il ruolo IAM Capability che fornisci determina quali AWS risorse ACK può creare e gestire. Per indicazioni sulla creazione di un ruolo appropriato con autorizzazioni con privilegi

minimi, consulta e. the section called "Capacità, ruolo IAM" the section called "Considerazioni sulle funzionalità EKS"

## Scegli il tuo strumento

È possibile creare una funzionalità ACK utilizzando Console di gestione AWS, AWS CLI o eksctl:

- the section called "Console"- Usa la console per un'esperienza guidata
- the section called " AWS CLI"- Usa la AWS CLI per lo scripting e l'automazione
- the section called "eksctl"- Usa eksctl per un'esperienza Kubernetes-native

## Cosa succede quando si crea una funzionalità ACK

Quando crei una funzionalità ACK:

1. EKS crea il servizio di funzionalità ACK e lo configura per monitorare e gestire le risorse nel cluster
2. Le definizioni delle risorse personalizzate (CRD) sono installate nel cluster
3. Viene creata automaticamente una voce di accesso per il tuo IAM Capability Role con policy di accesso specifiche per ogni funzionalità che concedono le autorizzazioni Kubernetes di base (vedi) the section called "Considerazioni sulle funzionalità EKS"
4. La funzionalità presuppone lo IAM Capability Role fornito dall'utente
5. ACK inizia a controllare le proprie risorse personalizzate nel cluster
6. Lo stato della capacità cambia da `CREATING` a `ACTIVE`

Una volta attivo, è possibile creare risorse ACK personalizzate nel cluster per gestire AWS le risorse.

> **Note**
>
> La voce di accesso creata automaticamente include la voce `AmazonEKSACKPolicy` che concede le autorizzazioni ACK per la gestione AWS delle risorse. Alcune risorse ACK che fanno riferimento ai segreti di Kubernetes (come i database RDS con password) richiedono politiche di accesso aggiuntive. Per ulteriori informazioni sulle voci di accesso e su come configurare autorizzazioni aggiuntive, consulta. the section called "Considerazioni sulle funzionalità EKS"

## Fasi successive

Dopo aver creato la funzionalità ACK:

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia a usare le AWS risorse
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM e i modelli multi-account

## Creare una funzionalità ACK utilizzando la console

Crea una funzionalità ACK sul tuo cluster Amazon EKS utilizzando la console AWS di gestione.

Crea la funzionalità ACK

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Nella barra di navigazione a sinistra, scegli  AWS Controllers for Kubernetes (ACK).
5. Scegli la funzionalità Crea AWS controller per Kubernetes.
6. Per IAM Capability Role:
   - Se disponi già di un IAM Capability Role, selezionalo dal menu a discesa
   - Se devi creare un ruolo, scegli Crea ruolo di amministratore

     Questo apre la console IAM in una nuova scheda con la policy di fiducia precompilata e la policy `AdministratorAccess` gestita. Se preferisci, puoi deselezionare questa policy e aggiungere altre autorizzazioni.

     Dopo aver creato il ruolo, torna alla console EKS e il ruolo verrà selezionato automaticamente.

     > **Important**
     >
     > La `AdministratorAccess` politica suggerita concede ampie autorizzazioni e ha lo scopo di semplificare l'avvio. Per uso in produzione, sostituiscila con una politica personalizzata che conceda solo le autorizzazioni necessarie per i AWS servizi specifici che intendi gestire con ACK. Per indicazioni sulla creazione di politiche con privilegi minimi, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

7. Scegli Create (Crea).

Inizia il processo di creazione delle capacità.

Verifica che la funzionalità sia attiva

1. Nella scheda Funzionalità, visualizza lo stato della capacità ACK.
2. Attendi che lo stato cambi da `CREATING` a`ACTIVE`.
3. Una volta attiva, la funzionalità è pronta per l'uso.

Per informazioni sullo stato delle funzionalità e sulla risoluzione dei problemi, vederethe section called "Lavorare con le funzionalità".

Verifica che siano disponibili risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate ACK siano disponibili nel cluster.

Utilizzo della console

1. Accedi al tuo cluster nella console Amazon EKS
2. Scegli la scheda Risorse
3. Scegli Estensioni
4. Scegliere CustomResourceDefinitions

Dovresti vedere un certo numero di CRD elencati per AWS le risorse.

Usare kubectl

```
kubectl api-resources | grep services.k8s.aws
```

Dovresti vedere un certo numero di API elencate per le risorse. AWS

> **Note**
>
> La funzionalità di AWS Controllers for Kubernetes consentirà di installare una serie di CRD per una varietà di risorse. AWS

Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM per altri servizi AWS
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità ACK

## Crea una funzionalità ACK utilizzando la AWS CLI

Crea una funzionalità ACK sul tuo cluster Amazon EKS utilizzando la AWS CLI. Questa procedura ti guida nella creazione di un ruolo IAM e nella verifica che la funzionalità sia attiva.

Prerequisiti

- AWS CLI: versione `2.12.3` o successiva. Per verificare la tua versione, `aws --version` esegui. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando.
- **`kubectl`**— Uno strumento da riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".

Fase 1: creazione di un ruolo IAM Capability

Crea un file di policy di fiducia:

```
cat > ack-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
```

```
}
EOF
```

Crea il ruolo IAM:

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://ack-trust-policy.json
```

Allega la policy `AdministratorAccess` gestita al ruolo:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

> **Important**
>
> La `AdministratorAccess` politica suggerita concede ampie autorizzazioni e ha lo scopo di semplificare l'avvio. Per uso in produzione, sostituiscila con una politica personalizzata che conceda solo le autorizzazioni necessarie per i AWS servizi specifici che intendi gestire con ACK. Per indicazioni sulla creazione di politiche con privilegi minimi, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

Fase 2: Creare la funzionalità ACK

Crea la risorsa di funzionalità ACK sul tuo cluster. Sostituiscila *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscila *`my-cluster`* con il nome del cluster.

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack \
  --type ACK \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ACKCapabilityRole \
  --delete-propagation-policy RETAIN
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva poiché EKS crea l'infrastruttura e i componenti di funzionalità richiesti. EKS installerà le Kubernetes Custom Resource Definitions relative a questa funzionalità nel cluster non appena viene creata.

> **Note**
>
> Se ricevi un errore che indica che il cluster non esiste o non disponi delle autorizzazioni, verifica:
>
> - Il nome del cluster è corretto
> - La tua AWS CLI è configurata per la regione corretta
> - Disponi delle autorizzazioni IAM richieste

Fase 3: Verifica che la funzionalità sia attiva

Attendi che la funzionalità diventi attiva. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscilo *`my-cluster`* con il nome del cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack \
  --query 'capability.status' \
  --output text
```

La funzionalità è pronta quando viene visualizzato lo stato`ACTIVE`. Non continuare con il passaggio successivo fino a quando lo stato non sarà raggiunto`ACTIVE`.

Puoi anche visualizzare i dettagli completi delle funzionalità:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack
```

Fase 4: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate ACK siano disponibili nel cluster:

```
kubectl api-resources | grep services.k8s.aws
```

Dovresti vedere un certo numero di API elencate per AWS le risorse.

> **Note**
>
> La funzionalità di AWS Controllers for Kubernetes consentirà di installare una serie di CRD per una varietà di risorse. AWS

Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM per altri servizi AWS
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità ACK

## Crea una funzionalità ACK usando eksctl

Crea una funzionalità ACK sul tuo cluster Amazon EKS usando eksctl.

> **Note**
>
> I passaggi seguenti richiedono la versione eksctl o successiva. `0.215.0` Per verificare la tua versione, esegui. `eksctl version`

Fase 1: Creare un ruolo di capacità IAM

Crea un file di policy di fiducia:

```
cat > ack-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
```

```
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crea il ruolo IAM:

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://ack-trust-policy.json
```

Allega la policy `AdministratorAccess` gestita al ruolo:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

> **Important**
>
> La `AdministratorAccess` politica suggerita concede ampie autorizzazioni e ha lo scopo di semplificare l'avvio. Per uso in produzione, sostituiscila con una politica personalizzata che conceda solo le autorizzazioni necessarie per i AWS servizi specifici che intendi gestire con ACK. Per indicazioni sulla creazione di politiche con privilegi minimi, consulta e. the section called “Configurazione delle autorizzazioni ” the section called “Considerazioni sulle funzionalità EKS”

Fase 2: Creare la funzionalità ACK

Crea la funzionalità ACK usando eksctl. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster e *`my-cluster`* sostituiscilo con il nome del cluster.

```
eksctl create capability \
  --cluster my-cluster \
  --region region-code \
```

```
  --name ack \
  --type ACK \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
text):role/ACKCapabilityRole
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva.

Fase 3: Verificare che la funzionalità sia attiva

Verifica lo stato della capacità:

```
eksctl get capability \
  --cluster my-cluster \
  --region region-code \
  --name ack
```

La funzionalità è pronta quando viene visualizzato lo stato`ACTIVE`.

Fase 4: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate ACK siano disponibili nel cluster:

```
kubectl api-resources | grep services.k8s.aws
```

Dovresti vedere un certo numero di API elencate per AWS le risorse.

> **Note**
>
> La funzionalità di AWS Controllers for Kubernetes consentirà di installare una serie di CRD per una varietà di risorse. AWS

Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM per altri servizi AWS
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità ACK

# Concetti ACK

ACK gestisce AWS le risorse tramite le API Kubernetes riconciliando continuamente lo stato desiderato nei tuoi manifest con lo stato effettivo in. AWS Quando crei o aggiorni una risorsa personalizzata Kubernetes, ACK effettua le chiamate AWS API necessarie per creare o modificare la AWS risorsa corrispondente, quindi ne monitora l'eventuale deriva e aggiorna lo stato di Kubernetes per riflettere lo stato corrente. Questo approccio consente di gestire l'infrastruttura utilizzando strumenti e flussi di lavoro Kubernetes familiari, mantenendo la coerenza tra il cluster e. AWS

Questo argomento spiega i concetti fondamentali alla base del modo in cui ACK gestisce le AWS risorse tramite le API Kubernetes.

## Guida introduttiva a ACK

Dopo aver creato la funzionalità ACK (vedithe section called "Crea funzionalità ACK"), puoi iniziare a gestire AWS le risorse utilizzando i manifest di Kubernetes nel tuo cluster.

Ad esempio, crea questo bucket manifest S3 in`bucket.yaml`, scegliendo il nome univoco del tuo bucket.

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-test-bucket
  namespace: default
spec:
  name: my-unique-bucket-name-12345
```

Applica il manifesto:

```
kubectl apply -f bucket.yaml
```

Controlla lo stato:

```
kubectl get bucket my-test-bucket
kubectl describe bucket my-test-bucket
```

Verifica che il bucket sia stato creato in AWS:

```
aws s3 ls | grep my-unique-bucket-name-12345
```

Elimina la risorsa Kubernetes:

```
kubectl delete bucket my-test-bucket
```

Verifica che il bucket sia stato eliminato da: AWS

```
aws s3 ls | grep my-unique-bucket-name-12345
```

Il bucket non dovrebbe più apparire nell'elenco, a dimostrazione che ACK gestisce l'intero ciclo di vita delle risorse. AWS

Per ulteriori informazioni su come iniziare con ACK, consulta Getting Started with ACK sul sito Web ACK.

## Ciclo di vita e riconciliazione delle risorse

ACK utilizza un ciclo di riconciliazione continuo per garantire che AWS le risorse corrispondano allo stato desiderato definito nei manifesti di Kubernetes.

Come funziona la riconciliazione:

1. Crei o aggiorni una risorsa personalizzata Kubernetes (ad esempio, un bucket S3)
2. ACK rileva la modifica e confronta lo stato desiderato con lo stato effettivo in AWS
3. Se differiscono, ACK effettua chiamate AWS API per riconciliare la differenza
4. ACK aggiorna lo stato delle risorse in Kubernetes per riflettere lo stato corrente
5. Il ciclo si ripete continuamente, in genere ogni poche ore

La riconciliazione viene attivata quando si crea una nuova risorsa Kubernetes, si aggiorna quella di una risorsa esistente o quando ACK rileva `spec` modifiche manuali apportate all'esterno di ACK. AWS Inoltre, ACK esegue una riconciliazione periodica con un periodo di risincronizzazione di 10 ore. Le modifiche alle risorse Kubernetes innescano una riconciliazione immediata, mentre il rilevamento passivo della deriva delle modifiche alle risorse a monte avviene durante la risincronizzazione periodica. AWS

Seguendo l'esempio introduttivo riportato sopra, ACK esegue questi passaggi:

1. Controlla se il bucket esiste in AWS
2. In caso contrario, chiama `s3:CreateBucket`
3. Aggiorna lo stato di Kubernetes con l'ARN e lo stato del bucket
4. Continua il monitoraggio della deriva

Per ulteriori informazioni su come funziona ACK, vedere ACK Reconciliation sul sito Web ACK.

## Condizioni relative allo stato

Le risorse ACK utilizzano le condizioni di stato per comunicare il proprio stato. La comprensione di queste condizioni aiuta a risolvere i problemi e a comprendere lo stato delle risorse.

- Pronta: indica che la risorsa è pronta per essere utilizzata (condizione Kubernetes standardizzata).
- ACK.ResourceSynced: indica che le specifiche della risorsa corrispondono allo stato della risorsa. AWS
- ACK.Terminal: indica che si è verificato un errore irreversibile.
- ACK.Adopted: indica che la risorsa è stata adottata da una AWS risorsa esistente anziché creata nuova.
- ACK.Recoverable: indica un errore recuperabile che può risolversi senza aggiornare le specifiche.
- ACK.Advisory: fornisce informazioni di consulenza sulla risorsa.
- ACK.LateInitialized: indica se l'inizializzazione tardiva dei campi è completa.
- ACK.ReferencesResolved: indica se tutti i `AWSResourceReference` campi sono stati risolti.
- ACK.IAMRoleSelected: indica se IAMRoleSelector è stato selezionato un per gestire questa risorsa.

Controlla lo stato della risorsa:

```
# Check if resource is ready
kubectl get bucket my-bucket -o jsonpath='{.status.conditions[?
(@.type=="Ready")].status}'

# Check for terminal errors
kubectl get bucket my-bucket -o jsonpath='{.status.conditions[?
(@.type=="ACK.Terminal")]}'
```

Stato di esempio:

```
status:
  conditions:
  - type: Ready
    status: "True"
    lastTransitionTime: "2024-01-15T10:30:00Z"
  - type: ACK.ResourceSynced
    status: "True"
    lastTransitionTime: "2024-01-15T10:30:00Z"
  - type: ACK.Terminal
    status: "False"
  ackResourceMetadata:
    arn: arn:aws:s3:::my-unique-bucket-name
    ownerAccountID: "111122223333"
    region: us-west-2
```

Per ulteriori informazioni sullo stato e sulle condizioni ACK, vedere Condizioni ACK sul sito Web ACK.

## Politiche di eliminazione

La politica di eliminazione di ACK controlla cosa succede alle AWS risorse quando si elimina la risorsa Kubernetes.

Elimina (impostazione predefinita)

La AWS risorsa viene eliminata quando si elimina la risorsa Kubernetes: questo è il comportamento predefinito.

```
# No annotation needed - this is the default
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: temp-bucket
spec:
  name: temporary-bucket
```

L'eliminazione di questa risorsa comporta l'eliminazione del bucket S3 in. AWS

Mantenimento

La AWS risorsa viene conservata quando si elimina la risorsa Kubernetes:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: important-bucket
  annotations:
    services.k8s.aws/deletion-policy: "retain"
spec:
  name: production-data-bucket
```

L'eliminazione di questa risorsa la rimuove da Kubernetes ma lascia il bucket S3 attivo. AWS

La `retain` policy è utile per i database di produzione che dovrebbero sopravvivere alla risorsa Kubernetes, alle risorse condivise utilizzate da più applicazioni, alle risorse con dati importanti che non devono essere eliminati accidentalmente o alla gestione temporanea degli ACK in cui si adotta una risorsa, la si configura e la si rilascia nuovamente alla gestione manuale.

Per ulteriori informazioni sulla politica di eliminazione degli ACK, consulta la Politica di eliminazione degli ACK sul sito Web ACK.

## Adozione delle risorse

L'adozione consente di sottoporre AWS le risorse esistenti alla gestione ACK senza ricrearle.

Quando utilizzare l'adozione:

- Migrazione dell'infrastruttura esistente alla gestione ACK
- Recupero di risorse orfane in caso di eliminazione accidentale delle AWS risorse in Kubernetes
- Importazione di risorse create da altri strumenti (, Terraform) CloudFormation

Come funziona l'adozione:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Quando crei questa risorsa:

1. ACK controlla se esiste un bucket con quel nome AWS
2. Se trovato, ACK lo adotta (nessuna chiamata API da creare)
3. ACK legge la configurazione corrente da AWS
4. ACK aggiorna lo stato di Kubernetes in base allo stato attuale
5. Gli aggiornamenti futuri riconciliano normalmente la risorsa

Una volta adottate, le risorse vengono gestite come qualsiasi altra risorsa ACK e l'eliminazione della risorsa Kubernetes eliminerà la risorsa a meno che non si utilizzi la politica AWS di eliminazione. `retain`

Quando si adottano le risorse, la AWS risorsa deve già esistere e ACK necessita delle autorizzazioni di lettura per scoprirla. La `adopt-or-create` policy adotta la risorsa se esiste o la crea se non esiste. Ciò è utile quando si desidera un flusso di lavoro dichiarativo che funzioni indipendentemente dal fatto che la risorsa esista o meno.

Per ulteriori informazioni sull'adozione delle risorse ACK, vedere ACK Resource Adoption sul sito Web ACK.

## Cross-account e risorse interregionali

ACK è in grado di gestire risorse in diversi AWS account e regioni da un singolo cluster.

Cross-region annotazioni sulle risorse

È possibile specificare la regione di una AWS risorsa utilizzando un'annotazione:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: eu-bucket
  annotations:
    services.k8s.aws/region: eu-west-1
spec:
  name: my-eu-bucket
```

Puoi anche specificare la regione di tutte le AWS risorse create in un determinato namespace:

Annotazioni relative allo spazio dei nomi

Imposta una regione predefinita per tutte le risorse in un namespace:

```
apiVersion: v1
kind: Namespace
metadata:
  name: production
  annotations:
    services.k8s.aws/default-region: us-west-2
```

Le risorse create in questo namespace utilizzano questa regione a meno che non vengano sovrascritte con un'annotazione a livello di risorsa.

Cross-account

Usa i selettori di ruolo IAM per mappare ruoli IAM specifici ai namespace:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: target-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
      - production
```

Le risorse create nel namespace mappato utilizzano automaticamente il ruolo specificato.

Per saperne di più sui selettori di ruolo IAM, vedi ACK Resource Management. Cross-Account
Per i dettagli sulla configurazione tra più account, consulta. the section called "Configurazione delle autorizzazioni "

## Gestione degli errori e comportamento dei tentativi

ACK gestisce automaticamente gli errori transitori e riprova le operazioni non riuscite.

Strategia di riprova:

- Gli errori transitori (limitazione della frequenza, problemi temporanei di servizio, autorizzazioni insufficienti) attivano nuovi tentativi automatici
- Il backoff esponenziale impedisce il sovraccarico delle API AWS
- Il numero massimo di tentativi di ripetizione varia in base al tipo di errore

- Gli errori permanenti (parametri non validi, conflitti tra i nomi delle risorse) non si riprovano

Controlla lo stato della risorsa per i dettagli degli errori utilizzando: `kubectl describe`

```
kubectl describe bucket my-bucket
```

Cerca le condizioni di stato con i messaggi di errore, gli eventi che mostrano i recenti tentativi di riconciliazione e il `message` campo nelle condizioni di stato che spiega gli errori. Gli errori più comuni includono autorizzazioni IAM insufficienti, conflitti tra i nomi delle risorse AWS, valori di configurazione non validi e quote di servizio superate`spec`. AWS

Per la risoluzione degli errori più comuni, consulta. the section called "Risoluzione dei problemi"

### Composizione delle risorse con kro

Per comporre e connettere più risorse ACK insieme, usa EKS Capability for kro (Kube Resource Orchestrator). kro fornisce un modo dichiarativo per definire gruppi di risorse, passando la configurazione tra le risorse per gestire semplicemente modelli di infrastruttura complessi.

Per esempi dettagliati sulla creazione di composizioni di risorse personalizzate con risorse ACK, vedi. the section called "concetti kro"

### Fasi successive

- the section called "Considerazioni"- EKS-specific modelli e strategie di integrazione

## Configurare le autorizzazioni ACK

ACK richiede le autorizzazioni IAM per creare e gestire AWS le risorse per tuo conto. Questo argomento spiega come IAM funziona con ACK e fornisce indicazioni sulla configurazione delle autorizzazioni per diversi casi d'uso.

### Come funziona IAM con ACK

ACK utilizza i ruoli IAM per autenticarsi AWS ed eseguire azioni sulle risorse. Esistono due modi per fornire le autorizzazioni ad ACK:

Ruolo di capacità: il ruolo IAM fornito durante la creazione della funzionalità ACK. Questo ruolo viene utilizzato di default per tutte le operazioni ACK.

Selettori di ruolo IAM: ruoli IAM aggiuntivi che possono essere mappati a namespace o risorse specifici. Questi ruoli sostituiscono il Capability Role per le risorse che rientrano nel loro ambito.

Quando ACK deve creare o gestire una risorsa, determina quale ruolo IAM utilizzare:

1. Controlla se an IAMRoleSelector corrisponde allo spazio dei nomi della risorsa
2. Se viene trovata una corrispondenza, assumi quel ruolo IAM
3. Altrimenti, usa il ruolo di capacità

Questo approccio consente una gestione flessibile delle autorizzazioni, da semplici configurazioni a ruolo singolo a configurazioni complesse con più account e più team.

## Guida introduttiva: semplice configurazione delle autorizzazioni

Per lo sviluppo, il test o i casi d'uso semplici, puoi aggiungere tutte le autorizzazioni di servizio necessarie direttamente al Capability Role.

Questo approccio funziona bene quando:

- Stai iniziando con ACK
- Tutte le risorse sono nello stesso AWS account
- Un unico team gestisce tutte le risorse ACK
- Ti fidi che tutti gli utenti ACK abbiano le stesse autorizzazioni

## Best practice di produzione: IAM Role Selectors

Per gli ambienti di produzione, utilizza i selettori di ruolo IAM per implementare l'accesso con privilegi minimi e l'isolamento a livello di namespace.

Quando si utilizzano i selettori di ruolo IAM, il ruolo di capacità necessita solo delle autorizzazioni necessarie per assumere i ruoli specifici del servizio. `sts:AssumeRole` `sts:TagSession` Non è necessario aggiungere alcuna autorizzazione di AWS servizio (come S3 o RDS) al Capability Role stesso: tali autorizzazioni vengono concesse ai singoli ruoli IAM assunti dal Capability Role.

Scelta tra modelli di autorizzazione:

Utilizza le autorizzazioni  dirette  (aggiungendo le autorizzazioni di servizio al Capability Role) quando:

- Stai iniziando e desideri la configurazione più semplice
- Tutte le risorse si trovano nello stesso account del cluster
- Hai requisiti di autorizzazione amministrativi a livello di cluster
- Tutti i team possono condividere le stesse autorizzazioni

Usa i selettori di ruolo IAM quando:

- Gestione delle risorse su più account AWS
- Team o namespace diversi richiedono autorizzazioni diverse
- È necessario un controllo granulare degli accessi per namespace
- Vuoi seguire le pratiche di sicurezza con privilegi minimi

Puoi iniziare con autorizzazioni dirette e migrare a IAM Role Selectors in un secondo momento man mano che le tue esigenze crescono.

Perché usare i selettori di ruolo IAM in produzione:

- Privilegio minimo: ogni namespace ottiene solo le autorizzazioni necessarie
- Isolamento del team: il team A non può utilizzare accidentalmente le autorizzazioni del team B
- Controllo più semplice: mappatura chiara di quale namespace utilizza quale ruolo
- Cross-account supporto: necessario per la gestione delle risorse in più account
- Separazione delle preoccupazioni: servizi o ambienti diversi utilizzano ruoli diversi

Configurazione di base di IAM Role Selector

Fase 1: Creare un ruolo IAM specifico per il servizio

Crea un ruolo IAM con autorizzazioni per servizi specifici: AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

Configura la policy di fiducia per consentire al Capability Role di assumerla:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": ["sts:AssumeRole", "sts:TagSession"]
    }
  ]
}
```

Fase 2: concedere AssumeRole l'autorizzazione a Capability Role

Aggiungi l'autorizzazione al Capability Role per assumere il ruolo specifico del servizio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::111122223333:role/ACK-S3-Role"
    }
  ]
}
```

Fase 3: Creazione IAMRoleSelector

Associa il ruolo IAM a un namespace:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
```

```
  name: s3-namespace-config
spec:
  arn: arn:aws:iam::111122223333:role/ACK-S3-Role
  namespaceSelector:
    names:
      - s3-resources
```

Passaggio 4: creare risorse nel namespace mappato

Le risorse nel `s3-resources` namespace utilizzano automaticamente il ruolo specificato:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: s3-resources
spec:
  name: my-production-bucket
```

## Multi-account gestione

Usa i selettori di ruolo IAM per gestire le risorse su più AWS account.

Fase 1: Creare un ruolo IAM per più account

Nell'account di destinazione (444455556666), crea un ruolo che consideri attendibile il ruolo di capacità dell'account di origine:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": ["sts:AssumeRole", "sts:TagSession"]
    }
  ]
}
```

Associa autorizzazioni specifiche del servizio a questo ruolo.

Passaggio 2: concedere l'autorizzazione AssumeRole

Nell'account di origine (111122223333), consenti al Capability Role di assumere il ruolo dell'account di destinazione:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::444455556666:role/ACKTargetAccountRole"
    }
  ]
}
```

Fase 3: Crea IAMRoleSelector

Associa il ruolo tra account a un namespace:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: production-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
      - production
```

Fase 4: Creare risorse

Le risorse nel `production` namespace vengono create nell'account di destinazione:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: production
spec:
  name: my-cross-account-bucket
```

## Tag di sessione

La funzionalità EKS ACK imposta automaticamente i tag di sessione su tutte le richieste AWS API. Questi tag consentono un controllo e un audit granulari degli accessi identificando l'origine di ogni richiesta.

Tag di sessione disponibili

I seguenti tag di sessione sono inclusi in ogni chiamata AWS API effettuata da ACK:

| Tag Key | Description |
| --- | --- |
| `eks:eks-capability-arn` | L'ARN della funzionalità EKS che effettua la richiesta |
| `eks:kubernetes-namespace` | Lo spazio dei nomi Kubernetes della risorsa gestita |
| `eks:kubernetes-api-group` | Il gruppo API Kubernetes della risorsa (ad esempio,) `s3.services.k8s.aws` |

Utilizzo dei tag di sessione per il controllo degli accessi

Puoi utilizzare questi tag di sessione nelle condizioni delle policy IAM per limitare le risorse che ACK può gestire. Ciò fornisce un ulteriore livello di sicurezza oltre ai selettori di ruolo IAM basati sullo spazio dei nomi.

Esempio: restrizione per namespace

Consenti ad ACK di creare bucket S3 solo quando la richiesta proviene dal namespace: `production`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:CreateBucket",
      "Resource": "*",
```

```
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks:kubernetes-namespace": "production"
        }
      }
    }
  ]
}
```

Esempio: restrizione in base alla capacità

Consenti azioni solo da una funzionalità ACK specifica:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks:eks-capability-arn": "arn:aws:eks:us-
west-2:111122223333:capability/my-cluster/ack/my-ack"
        }
      }
    }
  ]
}
```

> **Note**
>
> I tag di sessione sono diversi dall'ACK autogestito, che non imposta questi tag per impostazione predefinita. Ciò consente un controllo degli accessi più granulare con la funzionalità gestita.

## Schemi avanzati di selezione dei ruoli IAM

Per una configurazione avanzata, che include la mappatura dei ruoli tra account, la selezione dei ruoli con ambito di namespace ed esempi aggiuntivi, consulta Granular IAM Roles sul sito Web ACK.

### Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti di ACK e il ciclo di vita delle risorse
- the section called "Considerazioni sulle funzionalità EKS"- Comprendi le migliori pratiche di sicurezza per le funzionalità

## Considerazioni ACK per EKS

Questo argomento tratta importanti considerazioni sull'utilizzo di EKS Capability for ACK, tra cui la configurazione IAM, i modelli multi-account e l'integrazione con altre funzionalità EKS.

### Schemi di configurazione IAM

La funzionalità ACK utilizza un IAM Capability Role con AWS cui autenticarsi. Scegli il modello IAM giusto in base alle tue esigenze.

Semplice: ruolo con capacità singola

Per lo sviluppo, il test o i casi d'uso semplici, concedi tutte le autorizzazioni necessarie direttamente al Capability Role.

Quando usare:

- Guida introduttiva a ACK
- Single-account implementazioni
- Tutte le risorse gestite da un team
- Ambienti di sviluppo e test

Esempio: aggiungi le autorizzazioni S3 e RDS al tuo Capability Role con condizioni di etichettatura delle risorse:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:*"],
      "Resource": "*",
```

```
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-west-2", "us-east-1"]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["rds:*"],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-west-2", "us-east-1"],
          "aws:ResourceTag/ManagedBy": "ACK"
        }
      }
    }
  ]
}
```

Questo esempio limita le operazioni S3 e RDS a regioni specifiche e richiede che le risorse RDS abbiano un tag. `ManagedBy: ACK`

Produzione: IAM Role Selectors

Per gli ambienti di produzione, utilizza i selettori di ruolo IAM per implementare l'accesso con privilegi minimi e l'isolamento a livello di namespace.

Quando usare:

- Ambienti di produzione
- Multi-team cluster
- Multi-account gestione delle risorse
- Least-privilege requisiti di sicurezza
- Servizi diversi richiedono autorizzazioni diverse

Vantaggi:

- Ogni namespace ottiene solo le autorizzazioni necessarie
- Isolamento del team: il team A non può utilizzare le autorizzazioni del team B

- Controllo e conformità più semplici
- Necessario per la gestione delle risorse tra più account

Per una configurazione dettagliata di IAM Role Selector, consulta. the section called "Configurazione delle autorizzazioni "

## Integrazione con altre funzionalità EKS

### GitOps con Argo CD

Usa il CD EKS Capability for Argo per distribuire risorse ACK dai repository Git, abilitando i flussi di GitOps lavoro per la gestione dell'infrastruttura.

Considerazioni:

- Archivia le risorse ACK insieme ai manifesti delle applicazioni per un uso completo GitOps
- Organizza per ambiente, servizio o tipo di risorsa in base alla struttura del tuo team
- Usa la sincronizzazione automatica di Argo CD per una riconciliazione continua
- Abilita l'eliminazione per rimuovere automaticamente le risorse eliminate
- Prendi in considerazione i modelli hub-and-spoke per la gestione dell'infrastruttura multi-cluster

GitOps fornisce audit trail, funzionalità di rollback e gestione dichiarativa dell'infrastruttura. Per ulteriori informazioni su Argo CD, vedere. the section called "Lavorare con Argo CD"

### Composizione delle risorse con kro

Usa EKS Capability for kro (Kube Resource Orchestrator) per comporre più risorse ACK in astrazioni di livello superiore e API personalizzate.

Quando usare kro  con ACK:

- Crea modelli riutilizzabili per stack di infrastruttura comuni (database + backup + monitoraggio)
- Crea piattaforme self-service con API semplificate per i team applicativi
- Gestisci le dipendenze delle risorse e passa i valori tra le risorse (funzione da S3 bucket ARN a Lambda)
- Standardizza le configurazioni dell'infrastruttura tra i team
- Riduci la complessità nascondendo i dettagli di implementazione dietro risorse personalizzate

Schemi di esempio:

- Stack di applicazioni: bucket S3 + coda SQS + configurazione delle notifiche
- Configurazione del database: istanza RDS + gruppo di parametri + gruppo di sicurezza + segreti
- Rete: VPC + sottoreti + tabelle di routing + gruppi di sicurezza

kro gestisce l'ordinamento delle dipendenze, la propagazione dello stato e la gestione del ciclo di vita delle risorse composte. Per saperne di più su kro, vedi. the section called "concetti kro"

## Organizzazione delle risorse

Organizza le risorse ACK utilizzando i namespace e i tag AWS delle risorse Kubernetes per una migliore gestione, controllo degli accessi e monitoraggio dei costi.

### Organizzazione dei namespace

Usa i namespace Kubernetes per separare logicamente le risorse ACK per ambiente (produzione, staging, sviluppo), team (piattaforma, dati, ml) o applicazione.

Vantaggi:

- Namespace-scoped RBAC per il controllo degli accessi
- Imposta le regioni predefinite per namespace utilizzando le annotazioni
- Gestione e pulizia delle risorse più semplici
- Separazione logica allineata alla struttura organizzativa

### Aggiunta di tag alle risorse

La funzionalità EKS ACK applica automaticamente i tag predefiniti a tutte le AWS risorse che crea. Questi tag sono diversi dagli ACK autogestiti e forniscono una migliore tracciabilità.

Tag predefiniti applicati dalla funzionalità:

| Tag Key | Description |
| --- | --- |
| `eks:controller-version` | La versione del controller ACK |

| Tag Key | Description |
| --- | --- |
| `eks:kubernetes-namespace` | Lo spazio dei nomi Kubernetes della risorsa ACK |
| `eks:kubernetes-resource-name` | Il nome della risorsa Kubernetes |
| `eks:kubernetes-api-group` | Il gruppo di API Kubernetes (ad esempio,) `s3.services.k8s.aws` |
| `eks:eks-capability-arn` | L'ARN della funzionalità EKS ACK |

> **Note**
>
> Self-managed ACK utilizza diversi tag predefiniti: `services.k8s.aws/controller-version` `eservices.k8s.aws/namespace`. I tag della funzionalità utilizzano il `eks:` prefisso per garantire la coerenza con le altre funzionalità EKS.

Tag aggiuntivi consigliati:

Aggiungi tag personalizzati per l'allocazione dei costi, il monitoraggio della proprietà e per scopi organizzativi:

- Ambiente (produzione, messa in scena, sviluppo)
- Proprietà del team o del reparto
- Centro di costo per l'allocazione della fatturazione
- Nome dell'applicazione o del servizio

## Migrazione da altri Infrastructure-as-code strumenti

Molte organizzazioni stanno trovando valore nella standardizzazione su Kubernetes al di là dell'orchestrazione dei carichi di lavoro. La migrazione dell'infrastruttura e della gestione AWS delle risorse verso ACK consente di standardizzare la gestione dell'infrastruttura utilizzando le API Kubernetes insieme ai carichi di lavoro delle applicazioni.

Vantaggi della standardizzazione su Kubernetes per l'infrastruttura:

- Un'unica fonte di verità: gestisci sia le applicazioni che l'infrastruttura in Kubernetes, abilitando una pratica end-to-end GitOps
- Strumenti unificati: i team utilizzano le risorse e gli strumenti di Kubernetes anziché apprendere più strumenti e framework
- Riconciliazione coerente: ACK riconcilia continuamente le AWS risorse come fa Kubernetes per i carichi di lavoro, rilevando e correggendo la deriva rispetto agli strumenti imperativi
- Composizioni native: unendo kro e ACK, è possibile fare riferimento alle risorse direttamente nei manifesti delle applicazioni e delle risorse, passando stringhe di connessione e ARN tra AWS le risorse
- Operazioni semplificate: un unico piano di controllo per implementazioni, rollback e osservabilità sull'intero sistema

ACK supporta l'adozione AWS delle risorse esistenti senza ricrearle, consentendo una migrazione senza tempi di inattività da Terraform o da CloudFormation risorse esterne al cluster.

Adotta una  risorsa esistente:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Una volta adottata, la risorsa è gestita da ACK e può essere aggiornata tramite i manifesti di Kubernetes. È possibile eseguire la migrazione in modo incrementale, adottando le risorse necessarie e mantenendo gli strumenti IaC esistenti per altre risorse.

ACK supporta anche risorse di sola lettura. Per le risorse gestite da altri team o strumenti a cui vuoi fare riferimento ma non modificare, combina l'adozione con la politica di `retain` eliminazione e concedi solo le autorizzazioni IAM di lettura. Ciò consente alle applicazioni di scoprire l'infrastruttura condivisa (VPC, ruoli IAM, chiavi KMS) tramite le API Kubernetes senza rischiare modifiche.

Per ulteriori informazioni sull'adozione delle risorse, consulta. the section called “Concetti ACK”

## Politiche di eliminazione

Le politiche di eliminazione controllano cosa succede alle AWS risorse quando si elimina la risorsa Kubernetes corrispondente. Scegli la policy giusta in base al ciclo di vita delle risorse e ai tuoi requisiti operativi.

Elimina (impostazione predefinita)

La AWS risorsa viene eliminata quando si elimina la risorsa Kubernetes. Ciò mantiene la coerenza tra il cluster e AWS, assicurando che le risorse non si accumulino.

Quando usare delete:

- Ambienti di sviluppo e test in cui la pulizia è importante
- Risorse temporanee legate al ciclo di vita delle applicazioni (database di test, bucket temporanei)
- Risorse che non dovrebbero durare più a lungo dell'applicazione (code SQS, cluster) ElastiCache
- Ottimizzazione dei costi: ripulisce automaticamente le risorse inutilizzate
- Ambienti gestiti GitOps in cui la rimozione delle risorse da Git dovrebbe eliminare l'infrastruttura

La politica di eliminazione predefinita è in linea con il modello dichiarativo di Kubernetes: ciò che è nel cluster corrisponde a ciò che esiste in. AWS

Mantenimento

La AWS risorsa viene conservata quando si elimina la risorsa Kubernetes. Ciò protegge i dati critici e consente alle risorse di sopravvivere alla loro rappresentazione Kubernetes.

Quando usare keep:

- Database di produzione con dati critici che devono sopravvivere alle modifiche del cluster
- Long-term bucket di archiviazione con requisiti di conformità o di audit
- Risorse condivise utilizzate da più applicazioni o team
- Risorse in fase di migrazione verso diversi strumenti di gestione
- Scenari di disaster recovery in cui si desidera preservare l'infrastruttura
- Risorse con dipendenze complesse che richiedono un'attenta disattivazione

```
apiVersion: rds.services.k8s.aws/v1alpha1
```

```
kind: DBInstance
metadata:
  name: production-db
  annotations:
    services.k8s.aws/deletion-policy: "retain"
spec:
  dbInstanceIdentifier: prod-db
  # ... configuration
```

> **Important**
>
> Le risorse conservate continuano a comportare AWS costi e devono essere eliminate manualmente quando non sono più necessarie. AWS Usa l'etichettatura delle risorse per tenere traccia delle risorse conservate per la pulizia.

Per ulteriori informazioni sulle politiche di eliminazione, vedere. the section called "Concetti ACK"

## Documentazione upstream

Per informazioni dettagliate sull'uso di ACK, consulta le seguenti pagine del sito Web ACK:

- Per creare e gestire le risorse, consulta la guida all'uso di ACK.
- Per una documentazione completa sulle API per tutti i servizi, consulta il riferimento all'API ACK.
- Per una documentazione utente completa, consulta la documentazione ACK.

## Fasi successive

- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM e i modelli multi-account
- the section called "Concetti ACK"- Comprendi i concetti ACK e il ciclo di vita delle risorse
- the section called "Risoluzione dei problemi"- Risolvere i problemi relativi all'ACK
- the section called "Lavorare con Argo CD"- Implementa le risorse ACK con GitOps
- the section called "concetti kro"- Componi le risorse ACK in astrazioni di livello superiore

# Risolvi i problemi relativi alle funzionalità ACK

> **Note**
>
> Le funzionalità EKS sono completamente gestite ed eseguite all'esterno del cluster. Non hai accesso diretto ai namespace dei controller. È possibile configurare la distribuzione dei log del controller per la visibilità del comportamento del controller. Consulta the section called "Funzionalità, registri dei controller". La risoluzione dei problemi si concentra sullo stato delle funzionalità, sullo stato delle risorse e sulla configurazione IAM.

## La funzionalità è ATTIVA ma le risorse non vengono create

Se la tua funzionalità ACK mostra `ACTIVE` lo stato ma le risorse non vengono create in AWS, controlla lo stato della funzionalità, lo stato delle risorse e le autorizzazioni IAM.

Verifica lo stato della funzionalità:

È possibile visualizzare i problemi relativi all'integrità e allo stato delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la  scheda  Funzionalità per visualizzare lo stato e lo stato di tutte le funzionalità.

AWS CLI:

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack

# Look for issues in the health section
```

Cause comuni:

- Autorizzazioni IAM mancanti: al Capability Role mancano le autorizzazioni per il servizio AWS
- Namespace errato: risorse create nello spazio dei nomi senza che siano appropriate IAMRoleSelector
- Specifica della risorsa non valida: verifica la presenza di errori di convalida nelle condizioni di stato delle risorse
- Limitazione delle API: AWS i limiti di velocità delle API sono stati raggiunti
- Webhook di ammissione: webhook di ammissione che impediscono al controller di applicare patch allo stato delle risorse

Controlla lo stato delle risorse:

```
# Describe the resource to see conditions and events
kubectl describe bucket my-bucket -n default

# Look for status conditions
kubectl get bucket my-bucket -n default -o jsonpath='{.status.conditions}'

# View resource events
kubectl get events --field-selector involvedObject.name=my-bucket -n default
```

Verifica le autorizzazioni IAM:

```
# View the Capability Role's policies
aws iam list-attached-role-policies --role-name my-ack-capability-role
aws iam list-role-policies --role-name my-ack-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-ack-capability-role --policy-name policy-name
```

## Risorse create in AWS ma non viene visualizzato in Kubernetes

ACK tiene traccia solo delle risorse che crea tramite i manifesti di Kubernetes. Per gestire le AWS risorse esistenti con ACK, utilizza la funzionalità di adozione.

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
```

```
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Per ulteriori informazioni sull'adozione delle risorse, vederethe section called “Concetti ACK”.

## Cross-account risorse non create

Se le risorse non vengono create in un AWS account di destinazione quando si utilizzano i selettori di ruolo IAM, verifica la relazione di fiducia e la IAMRoleSelector configurazione.

Verifica la relazione  di fiducia:

```
# Check the trust policy in the target account role
aws iam get-role --role-name cross-account-ack-role --query
 'Role.AssumeRolePolicyDocument'
```

La politica di fiducia deve consentire al Capability Role dell'account di origine di assumerla.

Conferma IAMRoleSelector la configurazione:

```
# List IAMRoleSelectors (cluster-scoped)
kubectl get iamroleselector

# Describe specific selector
kubectl describe iamroleselector my-selector
```

Verifica l'allineamento dello spazio dei nomi:

IAMRoleSelectors sono risorse con ambito di cluster ma sono destinate a namespace specifici. Assicurati che le tue risorse ACK si trovino in un namespace che corrisponda al selettore dello spazio dei nomi: IAMRoleSelector

```
# Check resource namespace
kubectl get bucket my-cross-account-bucket -n production

# List all IAMRoleSelectors (cluster-scoped)
kubectl get iamroleselector
```

```
# Check which namespace the selector targets
kubectl get iamroleselector my-selector -o jsonpath='{.spec.namespaceSelector}'
```

Verifica la condizione: IAMRoleSelected

Verifica che sia IAMRoleSelector stata abbinata correttamente alla tua risorsa controllando la `ACK.IAMRoleSelected` condizione:

```
# Check if IAMRoleSelector was matched
kubectl get bucket my-cross-account-bucket -n production -o
 jsonpath='{.status.conditions[?(@.type=="ACK.IAMRoleSelected")]}'
```

Se la condizione è `False` o manca, il selettore IAMRoleSelector dello spazio dei nomi non corrisponde allo spazio dei nomi della risorsa. Verifica che il selettore `namespaceSelector` corrisponda alle etichette dello spazio dei nomi della risorsa.

Verifica le autorizzazioni del ruolo di capacità:

Requisiti `sts:AssumeRole` e `sts:TagSession` autorizzazioni del Capability Role per il ruolo dell'account di destinazione:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::444455556666:role/cross-account-ack-role"
    }
  ]
}
```

Per una configurazione dettagliata tra più account, vedere. the section called “Configurazione delle autorizzazioni ”

## Fasi successive

- the section called “Considerazioni”- Considerazioni e best practice relative all'ACK
- the section called “Configurazione delle autorizzazioni ”- Configurare le autorizzazioni IAM e i modelli multi-account

- the section called “Concetti ACK”- Comprendi i concetti ACK e il ciclo di vita delle risorse
- the section called “Funzionalità di risoluzione dei problemi”- Guida generale alla risoluzione dei problemi relativi alle funzionalità

## Confronto tra la capacità EKS per ACK e l'ACK autogestito

L'EKS Capability for ACK offre le stesse funzionalità dei controller ACK autogestiti, ma con vantaggi operativi significativi. Per un confronto generale tra EKS Capabilities e soluzioni autogestite, vedere. the section called “Considerazioni” Questo argomento si concentra sulle ACK-specific differenze.

### Differenze rispetto all'ACK upstream

L'EKS Capability for ACK si basa sui controller ACK upstream ma differisce nell'integrazione IAM.

Ruolo di capacità IAM: la funzionalità utilizza un ruolo IAM dedicato con una politica di fiducia che consente l'entità del `capabilities.eks.amazonaws.com` servizio, non l'IRSA (ruoli IAM per gli account di servizio). È possibile allegare le policy IAM direttamente al Capability Role senza dover creare o annotare gli account di servizio Kubernetes o configurare i provider OIDC. Una best practice per i casi d'uso in produzione consiste nel configurare le autorizzazioni del servizio utilizzando. `IAMRoleSelector` Per ulteriori dettagli, consulta the section called “Configurazione delle autorizzazioni ”.

Tag di sessione: la funzionalità gestita imposta automaticamente i tag di sessione su tutte le richieste AWS API, consentendo un controllo e un controllo granulari degli accessi. I tag includonoeks:eks-capability-arn,, e. eks:kubernetes-namespace eks:kubernetes-api-group Ciò è diverso dall'ACK autogestito, che non imposta questi tag per impostazione predefinita. the section called “Configurazione delle autorizzazioni ”Per informazioni dettagliate sull'uso dei tag di sessione nelle policy IAM, consulta.

Tag delle risorse: la funzionalità applica tag predefiniti diversi alle AWS risorse rispetto agli ACK autogestiti. La funzionalità utilizza tag con `eks:` prefisso (ad esempioeks:kubernetes-namespace,eks:eks-capability-arn) anziché i `services.k8s.aws/` tag utilizzati dall'ACK autogestito. the section called “Considerazioni”Per l'elenco completo dei tag delle risorse predefiniti, vedere.

Compatibilità delle risorse: le risorse personalizzate ACK funzionano in modo identico all'ACK originale senza modifiche ai file YAML delle risorse ACK. La funzionalità utilizza le stesse API e CRD di Kubernetes, quindi strumenti come questo funzionano allo stesso modo. `kubectl` La

funzionalità supporta solo i controller e le risorse generalmente disponibili (GA) nell'ACK upstream. La funzionalità non include i controller che sono in anteprima a monte. Lo stato di un controller può cambiare da anteprima a GA upstream nel tempo e la funzionalità può quindi iniziare a gestirlo automaticamente. Se oltre a questa funzionalità utilizzi un controller di anteprima autogestito, verificalo the section called "Controller di anteprima e promozione automatica" prima di eseguire la migrazione.

Per la documentazione completa di ACK e le guide specifiche sui servizi, consultate la documentazione ACK sul sito Web ACK.

## Percorso di migrazione

È possibile migrare dall'ACK autogestito alla funzionalità gestita con un'interruzione minima delle risorse. AWS La migrazione si basa sull'elezione del leader di Kubernetes: il controller autogestito e la funzionalità si contendono lo stesso contratto di locazione, quindi solo uno di essi riconcilia una determinata risorsa in qualsiasi momento. Affinché ciò funzioni, entrambi devono condividere un contratto di locazione nello stesso namespace. La funzionalità non richiede necessariamente il leasing di un controller autogestito funzionante, pertanto è possibile controllare quando avviene il passaggio di consegne ridimensionando il controller autogestito.

**Important**

Prima di iniziare, concedi allo IAM Capability Role le autorizzazioni equivalenti alle autorizzazioni utilizzate oggi dai controller autogestiti. La funzionalità si autentica con un Capability Role dedicato tramite il `capabilities.eks.amazonaws.com` service principal, anziché tramite il meccanismo attualmente utilizzato dai controller autogestiti, come IRSA o EKS Pod Identity (vedi). the section called "Configurazione delle autorizzazioni " Se al Capability Role mancano le autorizzazioni, la funzionalità adotta le tue risorse. Quindi non riesce a riconciliarle e registra gli errori. `AccessDenied`

Completa i passaggi seguenti per eseguire la migrazione. I passaggi utilizzano il controller S3 (`ack-s3-controller`) come esempio. Ripetili per ogni controller ACK autogestito che desideri migrare alla funzionalità, sostituendo il nome del controller e il diagramma Helm corrispondenti.

**Note**

L'esecuzione di un controller autogestito insieme alla funzionalità è intesa come uno stato temporaneo durante la migrazione, non come una configurazione a lungo termine. Mentre

entrambi sono in funzione, un'interruzione su entrambi i lati (ad esempio l'implementazione di una funzionalità o un aggiornamento del controller autogestito) può annullare il contratto di locazione e consentire all'altra parte di acquisirlo, provocando il passaggio inaspettato della riconciliazione tra i due. Completa la migrazione per ciascun controller anziché eseguirlo in modalità autogestita insieme alla funzionalità a tempo indeterminato.

1. Abilita l'elezione del leader sul tuo controller ACK autogestito e trasferisci il contratto di locazione a: `kube-system`

```
helm upgrade --install ack-s3-controller \
  oci://public.ecr.aws/aws-controllers-k8s/s3-chart \
  --namespace ack-system \
  --set leaderElection.enabled=true \
  --set leaderElection.namespace=kube-system
```

   È necessario impostare entrambi i valori. Nei grafici ACK Helm, il `--leader-election-namespace` flag viene applicato solo quando `leaderElection.enabled` lo è e l'elezione dei leader è disabilitata per impostazione predefinita. `true` L'impostazione `leaderElection.namespace` da sola non ha alcun effetto. Il controller continua a funzionare senza un contratto di leasing ed entrambi i controller riconciliano le stesse risorse contemporaneamente dopo la creazione della funzionalità. Questo vale per ogni grafico dei controller di servizio ACK, non solo per S3.

   Ciò sposta il contratto di locazione del controller a`kube-system`, consentendo alla capacità gestita di coordinarsi con esso.

2. Crea la funzionalità ACK sul tuo cluster (vedithe section called “Crea funzionalità ACK”). La funzionalità viene avviata e richiede il contratto di locazione, ma il controller autogestito la detiene comunque. Il controller autogestito continua a riconciliare le risorse e la capacità attende la leadership anziché forzare l'acquisizione.
3. Quando sei pronto per iniziare la migrazione, ridimensiona il controller autogestito fino a zero repliche. Questo rilascia il contratto di locazione in modo che la capacità possa acquisire la leadership e assumere il controllo della riconciliazione:

```
kubectl scale deployment ack-s3-controller \
  --namespace ack-system --replicas=0
```

Dopo aver ridimensionato il controller autogestito, la funzionalità acquisisce il contratto di locazione e inizia la riconciliazione, in genere entro breve tempo. La scalabilità del backup del controller autogestito non comporta la restituzione del contratto di leasing, poiché la funzionalità continua a mantenere e rinnovare il contratto di leasing. Per ripristinare la riconciliazione con il controller autogestito, ridimensionalo fino ad almeno una replica e quindi elimina la funzionalità ACK. Dopo aver eliminato la funzionalità, il controller autogestito riacquista il leasing e riprende la riconciliazione.

Una volta adottata, la funzionalità applica i propri tag di risorsa predefiniti (con `eks:` prefisso) al posto dei tag utilizzati dall'ACK autogestito (vedere). `services.k8s.aws/` the section called "Considerazioni" Aspettatevi un set una tantum di chiamate API di tagging alle risorse adottate e aggiornate qualsiasi strumento di allocazione dei costi o policy che inserisca il prefisso del tag. `services.k8s.aws/`

4. Verifica che la funzionalità sia valida e che abbia provveduto alla riconciliazione delle tue risorse. Verificate che le risorse segnalino una `Synced` condizione `True` e che la funzionalità non stia registrando errori. `AccessDenied`
5. Dopo aver verificato che la funzionalità sta gestendo correttamente le risorse, rimuovi il controller autogestito:

```
helm uninstall ack-s3-controller --namespace ack-system
```

Questo approccio consente a entrambi i controller di coesistere in sicurezza durante la migrazione. La funzionalità gestita adotta risorse precedentemente gestite da controller autogestiti dopo il rilascio del contratto di locazione, garantendo una riconciliazione continua senza conflitti.

## Controller di anteprima e promozione automatica

La funzionalità supporta solo i controller GA nell'ACK upstream. Un controller che è in anteprima oggi può essere promosso a GA upstream in un secondo momento. Quando ciò accade, la funzionalità inizia a gestire il controller automaticamente, senza alcuna azione da parte dell'utente.

Ciò comporta un rischio se si esegue un controller di anteprima autogestito su un cluster che utilizza la funzionalità anche per altri controller. È possibile eseguire il controller di anteprima come una singola replica con l'elezione del leader disattivata, poiché non c'è nessun altro controller con cui coordinarsi. Quando il controller viene promosso a GA, la funzionalità inizia a gestirlo. A quel punto, due riconciliatori agiscono sulle stesse risorse senza un contratto di locazione condiviso per

coordinarle. Il risultato è lo stesso conflitto di doppia riconciliazione che le fasi di migrazione sono progettate per prevenire. Entrambi i riconciliatori emettono chiamate AWS API concorrenti e scrivono aggiornamenti in conflitto sullo stato delle risorse personalizzate.

Per evitare questo problema, prima di eseguire un controller di anteprima autogestito insieme alla funzionalità:

- Attivate l'elezione del leader sul controller di anteprima autogestito e indicatene il leasing`kube-system`, utilizzando le stesse `leaderElection.namespace=kube-system` impostazioni `leaderElection.enabled=true` e le impostazioni mostrate nelle fasi di migrazione. In questo modo, se il controller viene promosso e la funzionalità lo sostituisce, i due sistemi si coordinino tramite un contratto di locazione condiviso anziché riconciliarsi in parallelo.
- Tieni traccia dello stato GA a monte di tutti i controller di anteprima da cui dipendi e pianifica di migrarli seguendo il percorso di migrazione quando verranno promossi. Puoi controllare lo stato corrente di ciascun controller nella pagina dei servizi ACK sul sito Web ACK.

### Fasi successive

- the section called “Crea funzionalità ACK”- Creare una risorsa di funzionalità ACK
- the section called “Concetti ACK”- Comprendere i concetti di ACK e il ciclo di vita delle risorse
- the section called “Configurazione delle autorizzazioni ”- Configurare IAM e autorizzazioni

## Implementazione continua con Argo CD

Argo CD è uno strumento dichiarativo e di distribuzione GitOps continua per Kubernetes. Con Argo CD, puoi automatizzare l'implementazione e la gestione del ciclo di vita delle tue applicazioni su più cluster e ambienti. Argo CD supporta diversi tipi di sorgenti tra cui repository Git, registri Helm (HTTP e OCI) e immagini OCI, offrendo flessibilità alle organizzazioni con diversi requisiti di sicurezza e conformità.

Con EKS Capabilities, Argo CD è completamente gestito da AWS, eliminando la necessità di installare, mantenere e scalare i controller Argo CD e le loro dipendenze dai cluster.

### Come funziona Argo CD

Argo CD segue GitOps lo schema, in cui la fonte dell'applicazione (repository Git, registro Helm o immagine OCI) è la fonte di verità per definire lo stato dell'applicazione desiderato. Quando si crea

una `Application` risorsa Argo CD, si specifica l'origine contenente i manifesti dell'applicazione e il cluster e lo spazio dei nomi Kubernetes di destinazione. Argo CD monitora continuamente sia la sorgente che lo stato live nel cluster, sincronizzando automaticamente qualsiasi modifica per garantire che lo stato del cluster corrisponda allo stato desiderato.

> **Note**
>
> Con EKS Capability for Argo CD, il software Argo CD viene eseguito nel piano di AWS controllo, non sui nodi di lavoro. Ciò significa che i nodi di lavoro non hanno bisogno dell'accesso diretto ai repository Git o ai registri Helm: la funzionalità gestisce l'accesso all'origine dall'account. AWS

Argo CD fornisce tre tipi di risorse principali:

- Applicazione: definisce una distribuzione da un repository Git a un cluster di destinazione
- ApplicationSet: genera più applicazioni da modelli per distribuzioni multi-cluster
- AppProject: Fornisce il raggruppamento logico e il controllo degli accessi per le applicazioni

Esempio: creazione di un'applicazione Argo CD

L'esempio seguente mostra come creare una risorsa Argo CD`Application`:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    name: in-cluster
    namespace: guestbook
  syncPolicy:
    automated:
      prune: true
```

```
    selfHeal: true
```

> **Note**
>
> Da utilizzare `destination.name` con il nome del cluster utilizzato durante la registrazione del cluster (come `in-cluster` per il cluster locale). Il `destination.server` campo funziona anche con gli ARN dei cluster EKS, ma si consiglia di utilizzare i nomi dei cluster per una migliore leggibilità.

## Vantaggi di Argo CD

Argo CD implementa un GitOps flusso di lavoro in cui si definiscono le configurazioni delle applicazioni nei repository Git e Argo CD sincronizza automaticamente le applicazioni in modo che corrispondano allo stato desiderato. Questo Git-centric approccio fornisce un audit trail completo di tutte le modifiche, consente un facile ripristino e si integra naturalmente con i processi di revisione e approvazione del codice esistenti. Argo CD rileva e riconcilia automaticamente la deriva tra lo stato desiderato in Git e lo stato effettivo nei cluster, assicurando che le distribuzioni rimangano coerenti con la configurazione dichiarata.

Con Argo CD, è possibile distribuire e gestire applicazioni su più cluster da una singola istanza Argo CD, semplificando le operazioni in ambienti multi-cluster e multiregione. L'interfaccia utente di Argo CD offre funzionalità di visualizzazione e monitoraggio, che consentono di visualizzare lo stato di implementazione, l'integrità e la cronologia delle applicazioni. L'interfaccia utente si integra con AWS Identity Center (precedentemente AWS SSO) per un'autenticazione e un'autorizzazione senza interruzioni, consentendoti di controllare l'accesso utilizzando l'infrastruttura di gestione delle identità esistente.

Come parte di EKS Managed Capabilities, Argo CD è completamente gestito da AWS, eliminando la necessità di installare, configurare e mantenere l'infrastruttura Argo CD. AWS gestisce la scalabilità, l'applicazione di patch e la gestione operativa, permettendo ai team di concentrarsi sulla distribuzione delle applicazioni piuttosto che sulla manutenzione degli strumenti.

## Integrazione con AWS Centro identità

EKS Managed Capabilities fornisce l'integrazione diretta tra Argo CD e AWS Identity Center, consentendo l'autenticazione e l'autorizzazione senza interruzioni per gli utenti. Quando abiliti la funzionalità Argo CD, puoi configurare l'integrazione di AWS Identity Center per mappare i gruppi e

gli utenti di Identity Center ai ruoli RBAC di Argo CD, consentendoti di controllare chi può accedere e gestire le applicazioni in Argo CD.

## Integrazione con altre funzionalità gestite da EKS

Argo CD si integra con altre funzionalità gestite da EKS.

- AWS Controller for Kubernetes (ACK): utilizza Argo CD per gestire l'implementazione delle risorse ACK su più cluster, abilitando i flussi di lavoro per la tua infrastruttura. GitOps AWS
- kro (Kube Resource Orchestrator): usa Argo CD per distribuire composizioni kro su più cluster, abilitando una composizione coerente delle risorse in tutto il tuo ambiente Kubernetes.

## Guida introduttiva ad Argo CD

Per iniziare a usare il CD EKS Capability for Argo:

1. Crea e configura un IAM Capability Role con le autorizzazioni necessarie per consentire ad Argo CD di accedere ai tuoi sorgenti e gestire le applicazioni.
2. Crea una risorsa con funzionalità Argo CD sul tuo cluster EKS tramite la AWS console, la AWS CLI o la tua infrastruttura preferita come strumento di codice.
3. Configura l'accesso al repository e registra i cluster per la distribuzione delle applicazioni.
4. Crea risorse applicative per distribuire le tue applicazioni dalle tue fonti dichiarative.

## Crea una funzionalità Argo CD

Questo argomento spiega come creare una funzionalità Argo CD sul tuo cluster Amazon EKS.

### Prerequisiti

Prima di creare una funzionalità Argo CD, assicurati di avere:

- Un cluster Amazon EKS esistente che esegue una versione di Kubernetes supportata (sono supportate tutte le versioni con supporto standard ed esteso)
- AWS Identity Center configurato: necessario per l'autenticazione Argo CD (gli utenti locali non sono supportati)
- Un ruolo di capacità IAM con autorizzazioni per Argo CD

- Autorizzazioni IAM sufficienti per creare risorse di funzionalità sui cluster EKS
- `kubectl`configurato per comunicare con il cluster
- (Opzionale) L'Argo CD CLI è stata installata per una più semplice gestione di cluster e repository
- (Per CLI/EksCtl) Lo strumento CLI appropriato installato e configurato

Per istruzioni sulla creazione dello IAM Capability Role, consulta. the section called "Capacità, ruolo IAM" Per la configurazione di Identity Center, consulta Guida introduttiva a AWS Identity Center.

> **Important**
>
> Il ruolo di capacità IAM fornito determina a quali AWS risorse Argo CD può accedere. Ciò include l'accesso al repository Git tramite CodeConnections e i segreti in Secrets Manager. Per indicazioni sulla creazione di un ruolo appropriato con autorizzazioni con privilegi minimi, consulta e. the section called "Capacità, ruolo IAM" the section called "Considerazioni sulle funzionalità EKS"

## Scegli il tuo strumento

È possibile creare una funzionalità Argo CD utilizzando la Console di gestione AWS AWS CLI o eksctl:

- the section called "Console"- Usa la console per un'esperienza guidata
- the section called " AWS CLI"- Usa la AWS CLI per lo scripting e l'automazione
- the section called "eksctl"- Usa eksctl per un'esperienza nativa di Kubernetes

## Cosa succede quando si crea una funzionalità Argo CD

Quando create una funzionalità Argo CD:

1. EKS crea il servizio di funzionalità Argo CD nel piano di controllo AWS
2. Le definizioni di risorse personalizzate (CRDs) sono installate nel cluster
3. Viene creata automaticamente una voce di accesso per il tuo IAM Capability Role con policy di accesso specifiche per ciascuna funzionalità che concedono le autorizzazioni Kubernetes di base (vedi) the section called "Considerazioni sulle funzionalità EKS"

4. Argo CD inizia a cercare le sue risorse personalizzate (Applicazioni,,) ApplicationSets AppProjects
5. Lo stato della funzionalità cambia da a `CREATING` `ACTIVE`
6. L'interfaccia utente di Argo CD diventa accessibile tramite il relativo URL

Una volta attiva, potete creare applicazioni Argo CD nel cluster da distribuire dalle vostre fonti dichiarative.

> **Note**
>
> La voce di accesso creata automaticamente non concede le autorizzazioni per distribuire applicazioni nei cluster. Per distribuire le applicazioni, è necessario configurare autorizzazioni Kubernetes RBAC aggiuntive per ogni cluster di destinazione. Vedi the section called "Registra i cluster" per i dettagli sulla registrazione dei cluster e sulla configurazione dell'accesso.

## Fasi successive

Dopo aver creato la funzionalità Argo CD:

- the section called "Concetti di Argo CD"- Scopri GitOps i principi, le politiche di sincronizzazione e i modelli multicluster
- the section called "Lavorare con Argo CD"- Configura l'accesso al repository, registra i cluster di destinazione e crea applicazioni
- the section called "Considerazioni su Argo CD"- Esplora i modelli di architettura multicluster e la configurazione avanzata

## Crea una funzionalità Argo CD utilizzando la console

Crea una funzionalità Argo CD sul tuo cluster Amazon EKS utilizzando la console di AWS gestione.

Prerequisiti

- AWS Identity Center configurato: Argo CD richiede AWS Identity Center per l'autenticazione. Gli utenti locali non sono supportati. Se non hai configurato AWS Identity Center, vedi Guida introduttiva a AWS Identity Center per creare un'istanza di Identity Center e Aggiungi utenti e Aggiungi gruppi per creare utenti e gruppi per l'accesso ad Argo CD.

- Almeno un utente o gruppo in AWS Identity Center: è necessario disporre di almeno un utente o gruppo configurato nell'istanza di Identity Center per assegnare le mappature dei ruoli RBAC di Argo CD e fornire l'accesso all'interfaccia utente di Argo CD.

### Crea la funzionalità Argo CD

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Nella barra di navigazione a sinistra, scegli Argo CD.
5. Scegli la funzionalità Create Argo CD.
6. Per il ruolo di capacità IAM:
   - Se disponi già di un ruolo di capacità IAM, selezionalo dal menu a discesa
   - Se devi creare un ruolo, scegli Create Argo CD role

   Si apre la console IAM in una nuova scheda con una politica di fiducia precompilata e l'accesso completo in lettura a Secrets Manager. Per impostazione predefinita, non vengono aggiunte altre autorizzazioni, ma puoi aggiungerle se necessario. Se prevedi di utilizzare CodeCommit repository o altri AWS servizi, aggiungi le autorizzazioni appropriate prima di creare il ruolo.

   Dopo aver creato il ruolo, torna alla console EKS e il ruolo verrà selezionato automaticamente.

   > **Note**
   >
   > Se prevedi di utilizzare le integrazioni opzionali con AWS Secrets Manager or AWS CodeConnections, dovrai aggiungere le autorizzazioni al ruolo. Per esempi di policy IAM e linee guida alla configurazione, consulta the section called " AWS Secrets Manager" e. the section called " AWS CodeConnections"

7. Configura AWS l'integrazione con Identity Center:
   a. Seleziona Abilita AWS l'integrazione con Identity Center.
   b. Scegli la tua istanza di Identity Center dal menu a discesa.
   c. Configura le mappature dei ruoli per RBAC assegnando utenti o gruppi ai ruoli Argo CD (ADMIN, EDITOR o VIEWER)
8. (Opzionale) Configura l'endpoint privato:

Per impostazione predefinita, l'interfaccia utente e l'endpoint API di Argo CD sono accessibili pubblicamente su Internet. Se è necessario limitare l'accesso, è possibile configurare un endpoint VPC. Questo è consigliato per ambienti con severi requisiti di sicurezza di rete.

a. Prima di creare la funzionalità, crea un endpoint VPC di interfaccia per il `com.amazonaws.<region>.eks-capabilities` servizio nel tuo VPC. L'endpoint VPC dovrebbe:
   - Essere associato a sottoreti in diverse zone di disponibilità per un'elevata disponibilità
   - Disponi di un gruppo di sicurezza che consenta il traffico HTTPS (porta 443) in entrata dalle reti che devono accedere all'interfaccia utente e all'API di Argo CD
   - Per maggiori dettagli sulla creazione e la personalizzazione degli endpoint VPC, consulta Create a VPC endpoint nella Guida. AWS PrivateLink

b. Nella sezione Argo CD endpoint access - opzionale della pagina di creazione della funzionalità Argo CD, seleziona Privato.

c. Scegli l'endpoint VPC che hai creato dal menu a discesa.

> **Note**
>
> Quando l'endpoint privato è abilitato, l'interfaccia utente e l'API di Argo CD sono accessibili solo tramite l'endpoint VPC. Gli utenti devono essere connessi al VPC (o a una rete peered) per accedere all'interfaccia Argo CD.

9. Scegli Crea.

Inizia il processo di creazione delle capacità.

Verificare che la funzionalità sia attiva

1. Nella scheda Funzionalità, visualizza lo stato della funzionalità del CD Argo.
2. Attendi che lo stato cambi da `CREATING` a`ACTIVE`.
3. Una volta attiva, la funzionalità è pronta per l'uso.

Per informazioni sullo stato delle funzionalità e sulla risoluzione dei problemi, vedere. the section called "Lavorare con le funzionalità"

Accedere all'interfaccia utente del CD di Argo

Dopo che la funzionalità è attiva, puoi accedere all'interfaccia utente di Argo CD:

1. Nella pagina delle funzionalità di Argo CD, scegli  Open Argo CD UI.
2. L'interfaccia utente di Argo CD si apre in una nuova scheda del browser.
3. È ora possibile creare applicazioni e gestire le distribuzioni tramite l'interfaccia utente.

Fasi successive

- the section called "Lavorare con Argo CD"- Configura i repository, registra i cluster e crea applicazioni
- the section called "Considerazioni su Argo CD"- Multi-cluster architettura e configurazione avanzata
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità Argo CD

## Crea una funzionalità Argo CD usando AWS CLI

Crea una funzionalità Argo CD sul tuo cluster Amazon EKS utilizzando la CLI AWS . Questa procedura ti guida nella creazione di un ruolo IAM, nella configurazione AWS dell'integrazione con Identity Center e nella verifica che la funzionalità sia attiva.

Prerequisiti

- AWS CLI: versione `2.12.3` o successiva. Per verificare la tua versione, `aws --version` esegui. Per ulteriori informazioni, vedere la Guida per l'utente all'installazione nella AWS Command Line Interface.
- **`kubectl`**— Uno strumento a riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".
- AWS Identity Center configurato: Argo CD richiede AWS Identity Center per l'autenticazione. Gli utenti locali non sono supportati. Se non hai configurato AWS Identity Center, vedi Guida introduttiva a AWS Identity Center per creare un'istanza di Identity Center e Aggiungi utenti e Aggiungi gruppi per creare utenti e gruppi per l'accesso ad Argo CD.
- Almeno un utente o gruppo in AWS Identity Center: è necessario disporre di almeno un utente o gruppo configurato nell'istanza di Identity Center per assegnare le mappature dei ruoli RBAC di Argo CD e fornire l'accesso all'interfaccia utente di Argo CD.

Fase 1: Creare un ruolo di capacità IAM

Crea un file di policy sulla fiducia:

```
cat > argocd-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crea il ruolo IAM:

```
aws iam create-role \
  --role-name ArgoCDCapabilityRole \
  --assume-role-policy-document file://argocd-trust-policy.json
```

> **Note**
>
> Se prevedi di utilizzare le integrazioni opzionali con AWS Secrets Manager o AWS CodeConnections, aggiungi le autorizzazioni al ruolo. Per esempi di policy IAM e linee guida alla configurazione, consulta the section called " AWS Secrets Manager" e. the section called " AWS CodeConnections"

(Facoltativo) Configura un endpoint privato

Per impostazione predefinita, l'interfaccia utente e l'endpoint API di Argo CD sono accessibili pubblicamente su Internet. Se è necessario limitare l'accesso, è possibile configurare un endpoint VPC. Questo è consigliato per ambienti con severi requisiti di sicurezza di rete.

Crea un endpoint VPC per EKS Capabilities

Crea un endpoint VPC di interfaccia per il servizio EKS Capabilities nel tuo VPC. Sostituisci*vpc-id*, *subnet-id-1 subnet-id-2sg-id*, e *region-code* con i tuoi valori:

```
aws ec2 create-vpc-endpoint \
  --vpc-endpoint-type Interface \
  --service-name com.amazonaws.region-code.eks-capabilities \
  --vpc-id vpc-id \
  --subnet-ids subnet-id-1 subnet-id-2 \
  --security-group-ids sg-id \
  --region region-code
```

**Note**

- Le sottoreti devono trovarsi in zone di disponibilità diverse per garantire l'alta disponibilità.
- Il gruppo di sicurezza deve consentire il traffico HTTPS (porta 443) in entrata dalle reti che devono accedere all'interfaccia utente e all'API di Argo CD.
- Nota l'ID dell'endpoint VPC restituito da questo comando: ti servirà durante la creazione della funzionalità.

Verifica che l'endpoint VPC sia disponibile

```
aws ec2 describe-vpc-endpoints \
  --vpc-endpoint-ids vpce-xxxxxxxx \
  --query 'VpcEndpoints[0].State' \
  --output text \
  --region region-code
```

Attendi che venga visualizzato lo stato `available` prima di procedere.

Fase 2: Creare la funzionalità Argo CD

Crea la risorsa di funzionalità Argo CD sul tuo cluster.

Innanzitutto, imposta le variabili di ambiente per la configurazione del tuo Identity Center:

```
# Get your Identity Center instance ARN (replace region if your IDC instance is in a
 different region)
export IDC_INSTANCE_ARN=$(aws sso-admin list-instances --region region-code --query
 'Instances[0].InstanceArn' --output text)

# Get a user ID for RBAC mapping (replace with your username and region if needed)
export IDC_USER_ID=$(aws identitystore list-users \
  --region region-code \
  --identity-store-id $(aws sso-admin list-instances --region region-code --query
 'Instances[0].IdentityStoreId' --output text) \
  --query 'Users[?UserName==your-username].UserId' --output text)

echo "IDC_INSTANCE_ARN=$IDC_INSTANCE_ARN"
echo "IDC_USER_ID=$IDC_USER_ID"
```

Crea la funzionalità con l'integrazione di Identity Center. Sostituiscila *region-code* con la AWS regione in cui si trova il cluster, *my-cluster* con il nome del cluster e *idc-region-code* con il codice regionale in cui si trova l'istanza di Identity Center:

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --type ARGOCD \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ArgoCDCapabilityRole \
  --delete-propagation-policy RETAIN \
  --configuration '{
    "argoCd": {
      "awsIdc": {
        "idcInstanceArn": "'$IDC_INSTANCE_ARN'",
        "idcRegion": "'idc-region-code'"
      },
      "rbacRoleMappings": [{
        "role": "ADMIN",
        "identities": [{
          "id": "'$IDC_USER_ID'",
          "type": "SSO_USER"
        }]
      }]
    }
  }'
```

Se hai configurato un endpoint VPC per l'accesso privato, includi il `network-configuration` parametro per creare la funzionalità con un endpoint privato. Sostituiscilo *vpce-xxxxxxxx* con il tuo ID endpoint VPC:

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --type ARGOCD \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ArgoCDCapabilityRole \
  --delete-propagation-policy RETAIN \
  --network-configuration '{
    "elasticNetworkInterfaces": {
      "vpcEndpointId": "'vpce-xxxxxxxx'"
    }
  }' \
  --configuration '{
    "argoCd": {
      "awsIdc": {
        "idcInstanceArn": "'$IDC_INSTANCE_ARN'",
        "idcRegion": "'idc-region-code'"
      },
      "rbacRoleMappings": [{
        "role": "ADMIN",
        "identities": [{
          "id": "'$IDC_USER_ID'",
          "type": "SSO_USER"
        }]
      }]
    }
  }'
```

Il comando ritorna immediatamente, ma la funzionalità impiega del tempo per diventare attiva poiché EKS crea l'infrastruttura e i componenti di funzionalità richiesti. EKS installa le Kubernetes Custom Resource Definitions (CRD) nel cluster durante la creazione delle funzionalità.

> **Note**
>
> Se ricevi un errore indicante che il cluster non esiste o non disponi delle autorizzazioni, verifica:

- Il nome del cluster è corretto
- La tua AWS CLI è configurata per la regione corretta
- Disponi delle autorizzazioni IAM richieste

Passaggio 3: verifica che la funzionalità sia attiva

Attendi che la funzionalità diventi attiva. Sostituisci *region-code* con la AWS regione in cui si trova il cluster e *my-cluster* con il nome del cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.status' \
  --output text
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE. Attendi che lo stato sia ACTIVE impostato prima di continuare con la fase successiva.

Puoi anche visualizzare i dettagli completi delle funzionalità:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd
```

Passaggio 4: verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate di Argo CD siano disponibili nel tuo cluster:

```
kubectl api-resources | grep argoproj.io
```

Dovresti vedere i tipi Application di ApplicationSet risorse elencati.

Fasi successive

- the section called "Lavorare con Argo CD"- Configura i repository, registra i cluster e crea applicazioni

- the section called "Considerazioni su Argo CD"- Multi-cluster architettura e configurazione avanzata
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità Argo CD

## Crea una funzionalità Argo CD usando eksctl

Crea una funzionalità Argo CD sul tuo cluster Amazon EKS usando eksctl.

> **Note**
>
> I passaggi seguenti richiedono una versione eksctl o successiva. `0.215.0` Per verificare la tua versione, esegui. `eksctl version`

Prerequisiti

- AWS Identity Center configurato: Argo CD richiede AWS Identity Center per l'autenticazione. Gli utenti locali non sono supportati. Se non hai configurato AWS Identity Center, vedi Guida introduttiva a AWS Identity Center per creare un'istanza di Identity Center e Aggiungi utenti e Aggiungi gruppi per creare utenti e gruppi per l'accesso ad Argo CD.
- Almeno un utente o gruppo in AWS Identity Center: è necessario disporre di almeno un utente o gruppo configurato nell'istanza di Identity Center per assegnare le mappature dei ruoli RBAC di Argo CD e fornire l'accesso all'interfaccia utente di Argo CD.

Fase 1: Creazione di un ruolo IAM Capability

Crea un file di policy sulla fiducia:

```
cat > argocd-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
```

```
      ]
    }
  ]
}
EOF
```

Crea il ruolo IAM:

```
aws iam create-role \
  --role-name ArgoCDCapabilityRole \
  --assume-role-policy-document file://argocd-trust-policy.json
```

> **Note**
>
> Per questa configurazione di base, non sono necessarie politiche IAM aggiuntive. Se prevedi di utilizzare Secrets Manager per le credenziali del repository o CodeConnections, dovrai aggiungere le autorizzazioni al ruolo. Per esempi di policy IAM e linee guida alla configurazione, consulta e. the section called " AWS Secrets Manager" the section called " AWS CodeConnections"

Fase 2: Ottieni il tuo AWS Configurazione dell'Identity Center

Ottieni l'ARN e l'ID utente dell'istanza di Identity Center per la configurazione RBAC:

```
# Get your Identity Center instance ARN
aws sso-admin list-instances --query 'Instances[0].InstanceArn' --output text

# Get a user ID for admin access (replace 'your-username' with your Identity Center
 username)
aws identitystore list-users \
  --identity-store-id $(aws sso-admin list-instances --query
 'Instances[0].IdentityStoreId' --output text) \
  --query 'Users[?UserName==`your-username`].UserId' --output text
```

Annota questi valori: ti serviranno nel passaggio successivo.

(Facoltativo) Configura un endpoint privato

Per impostazione predefinita, l'interfaccia utente e l'endpoint API di Argo CD sono accessibili pubblicamente su Internet. Se è necessario limitare l'accesso, è possibile configurare un endpoint VPC. Questo è consigliato per ambienti con severi requisiti di sicurezza di rete.

Crea un endpoint VPC per EKS Capabilities

Crea un endpoint VPC di interfaccia per il servizio EKS Capabilities nel tuo VPC. Sostituisci*vpc-id*, *subnet-id-1 subnet-id-2sg-id*, e *region-code* con i tuoi valori:

```
aws ec2 create-vpc-endpoint \
  --vpc-endpoint-type Interface \
  --service-name com.amazonaws.region-code.eks-capabilities \
  --vpc-id vpc-id \
  --subnet-ids subnet-id-1 subnet-id-2 \
  --security-group-ids sg-id \
  --region region-code
```

**Note**

- Le sottoreti devono trovarsi in zone di disponibilità diverse per garantire l'alta disponibilità.
- Il gruppo di sicurezza deve consentire il traffico HTTPS (porta 443) in entrata dalle reti che devono accedere all'interfaccia utente e all'API di Argo CD.
- Nota l'ID dell'endpoint VPC restituito da questo comando: ti servirà durante la creazione della funzionalità.

Verifica che l'endpoint VPC sia disponibile

```
aws ec2 describe-vpc-endpoints \
  --vpc-endpoint-ids vpce-xxxxxxxx \
  --query 'VpcEndpoints[0].State' \
  --output text \
  --region region-code
```

Attendi che venga visualizzato lo stato `available` prima di procedere.

Passo 3: Creare un file di configurazione eksctl

Crea un archivio denominato `argocd-capability.yaml` con i seguenti contenuti. Sostituisci i valori segnaposto con il nome del cluster, la regione del cluster, l'ARN del ruolo IAM, l'ARN dell'istanza di Identity Center, la regione dell'Identity Center e l'ID utente:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: cluster-region-code

capabilities:
  - name: my-argocd
    type: ARGOCD
    roleArn: arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
    deletePropagationPolicy: RETAIN
    configuration:
      argocd:
        awsIdc:
          idcInstanceArn: arn:aws:sso:::instance/ssoins-123abc
          idcRegion: idc-region-code
        rbacRoleMappings:
          - role: ADMIN
            identities:
              - id: 38414300-1041-708a-01af-5422d6091e34
                type: SSO_USER
```

> **Note**
>
> È possibile aggiungere più utenti o gruppi alle mappature RBAC. Per i gruppi, utilizza `type: SSO_GROUP` e fornisci l'ID del gruppo. I ruoli disponibili sono`ADMIN`,`EDITOR`, e`VIEWER`.

Se hai configurato un endpoint VPC per l'accesso privato, aggiungi la `networkConfiguration` sezione alla definizione della funzionalità. Sostituisci *`vpce-xxxxxxxx`* con il tuo ID endpoint VPC:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
```

```
  name: my-cluster
  region: cluster-region-code

capabilities:
  - name: my-argocd
    type: ARGOCD
    roleArn: arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
    deletePropagationPolicy: RETAIN
    networkConfiguration:
      elasticNetworkInterfaces:
        vpcEndpointId: vpce-xxxxxxxx
    configuration:
      argocd:
        awsIdc:
          idcInstanceArn: arn:aws:sso:::instance/ssoins-123abc
          idcRegion: idc-region-code
        rbacRoleMappings:
          - role: ADMIN
            identities:
              - id: 38414300-1041-708a-01af-5422d6091e34
                type: SSO_USER
```

> **Note**
>
> Quando l'endpoint privato è abilitato, l'interfaccia utente e l'API di Argo CD sono accessibili solo tramite l'endpoint VPC. Gli utenti devono essere connessi al VPC (o a una rete peered) per accedere all'interfaccia Argo CD.

Fase 4: Creare la funzionalità Argo CD

Applica il file di configurazione:

```
eksctl create capability -f argocd-capability.yaml
```

Il comando viene restituito immediatamente, ma la funzionalità richiede del tempo per diventare attiva.

Fase 5: Verificare che la funzionalità sia attiva

Controlla lo stato della capacità. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e sostituiscilo *my-cluster* con il nome del cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
  --name my-argocd
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE.

Passaggio 6: verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate di Argo CD siano disponibili nel tuo cluster:

```
kubectl api-resources | grep argoproj.io
```

Dovresti vedere i tipi Application di ApplicationSet risorse elencati.

Fasi successive

- the section called "Lavorare con Argo CD"- Scopri come creare e gestire le applicazioni Argo CD
- the section called "Considerazioni su Argo CD"- Configura l'accesso SSO e multi-cluster
- the section called "Lavorare con le funzionalità"- Gestisci le tue risorse di funzionalità Argo CD

## Concetti di Argo CD

Argo CD implementa GitOps trattando Git come l'unica fonte di verità per le implementazioni delle applicazioni. Questo argomento illustra un esempio pratico, quindi spiega i concetti fondamentali da comprendere quando si lavora con il CD EKS Capability for Argo.

### Guida introduttiva ad Argo CD

Dopo aver creato la funzionalità Argo CD (vedithe section called "Crea una funzionalità Argo CD"), puoi iniziare a distribuire le applicazioni. Questo esempio illustra la registrazione di un cluster e la creazione di un'applicazione.

Fase 1: configurazione

Registra il tuo cluster (richiesto)

Registra il cluster in cui desideri distribuire le applicazioni. Per questo esempio, registreremo lo stesso cluster su cui è in esecuzione Argo CD (potete usare il nome `in-cluster` per motivi di compatibilità con la maggior parte degli esempi di Argo CD):

```
# Get your cluster ARN
CLUSTER_ARN=$(aws eks describe-cluster \
  --name my-cluster \
  --query 'cluster.arn' \
  --output text)

# Register the cluster using Argo CD CLI
argocd cluster add $CLUSTER_ARN \
  --aws-cluster-name $CLUSTER_ARN \
  --name in-cluster \
  --project default
```

> **Note**
>
> Per informazioni sulla configurazione dell'Argo CD CLI per funzionare con la funzionalità Argo CD in EKS, vedere. the section called "Utilizzo dell'Argo CD CLI con la funzionalità gestita"

In alternativa, registra il cluster utilizzando un Kubernetes Secret (vedi per i dettagli). the section called "Registra i cluster"

Configura l'accesso al repository (opzionale)

Questo esempio utilizza un GitHub archivio pubblico, quindi non è richiesta alcuna configurazione del repository. Per gli archivi privati, configura l'accesso utilizzando AWS Secrets Manager o Kubernetes Secrets (vedi the section called "Configura i repository" per i dettagli). CodeConnections

Per AWS i servizi (grafici ECR for Helm e CodeCommit) CodeConnections, puoi farvi riferimento direttamente nelle risorse dell'applicazione senza creare un repository. Il Capability Role deve disporre delle autorizzazioni IAM richieste. Per informazioni dettagliate, vedi the section called "Configura i repository".

Fase 2: crea un'applicazione

Crea questo manifesto dell'applicazione in`my-app.yaml`:

```
apiVersion: argoproj.io/v1alpha1
```

```
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    name: in-cluster
    namespace: guestbook
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=true
```

Applica l'applicazione:

```
kubectl apply -f my-app.yaml
```

Dopo aver applicato questa applicazione, Argo CD: 1. Sincronizza l'applicazione da Git al tuo cluster (distribuzione iniziale) 2. Monitora il repository Git per le modifiche 3. Sincronizza automaticamente le modifiche successive al cluster 4. Rileva e corregge qualsiasi deriva dallo stato desiderato 5. Fornisce lo stato di salute e la cronologia di sincronizzazione nell'interfaccia utente

Visualizza lo stato dell'applicazione:

```
kubectl get application guestbook -n argocd
```

È inoltre possibile visualizzare l'applicazione utilizzando la CLI di Argo CD o l'interfaccia utente di Argo CD (accessibile dalla console EKS nella scheda Capacità del cluster).

> **Note**
>
> Quando si utilizza l'Argo CD CLI con la funzionalità gestita, specificare le applicazioni con il prefisso dello spazio dei nomi:. `argocd app get argocd/guestbook`

> **Note**
>
> Usa il nome del cluster in `destination.name` (il nome che hai usato durante la registrazione del cluster). La funzionalità gestita non supporta l'impostazione predefinita locale all'interno del cluster ()`kubernetes.default.svc`.

## Concetti principali

GitOps principi e tipi di fonti

Argo CD implementa GitOps, in cui la fonte dell'applicazione è l'unica fonte attendibile per le implementazioni:

- Dichiarativo: lo stato desiderato viene dichiarato utilizzando i manifesti YAML, i grafici Helm o gli overlay Kustomize
- Versionato: ogni modifica viene tracciata con un audit trail completo
- Automatizzato: Argo CD monitora continuamente le sorgenti e sincronizza automaticamente le modifiche
- Riparazione automatica: rileva e corregge lo scostamento tra lo stato desiderato e quello effettivo del cluster

Tipi di sorgenti supportati:

- Archivi Git - GitHub GitLab, Bitbucket, CodeCommit (HTTPS, SSH o) CodeConnections
- Registri Helm: registri HTTP (simili) e registri OCI (simili) `https://aws.github.io/eks-charts` `public.ecr.aws`
- Immagini OCI: immagini dei contenitori contenenti manifesti o grafici Helm (come) `oci://registry-1.docker.io/user/my-app`

Questa flessibilità consente alle organizzazioni di scegliere fonti che soddisfino i propri requisiti di sicurezza e conformità. Ad esempio, le organizzazioni che limitano l'accesso a Git dai cluster possono utilizzare ECR per i grafici Helm o le immagini OCI.

Per ulteriori informazioni, consultate Application Sources nella documentazione del CD Argo.

### Sincronizzazione e riconciliazione

Argo CD monitora continuamente le sorgenti e i cluster per rilevare e correggere le differenze:

1. Esamina le fonti delle modifiche (impostazione predefinita: ogni 6 minuti)
2. Confronta lo stato desiderato con lo stato del cluster
3. Contrassegna le applicazioni come o `Synced` `OutOfSync`
4. Sincronizza automaticamente le modifiche (se configurate) o attende l'approvazione manuale
5. Monitora lo stato delle risorse dopo la sincronizzazione

Le onde di sincronizzazione controllano l'ordine di creazione delle risorse utilizzando le annotazioni:

```
metadata:
  annotations:
    argocd.argoproj.io/sync-wave: "0"  # Default if not specified
```

Le risorse vengono applicate in ordine ondulatorio (partendo dai numeri più bassi, inclusi i numeri negativi come`-1`). Wave `0` è l'impostazione predefinita se non specificata. Ciò consente di creare dipendenze come i namespace (wave) prima delle distribuzioni (wave`-1`) prima dei servizi (wave`0`). 1

La riparazione automatica annulla automaticamente le modifiche manuali:

```
spec:
  syncPolicy:
    automated:
      selfHeal: true
```

> **Note**
>
> La funzionalità gestita utilizza il tracciamento delle risorse basato sulle annotazioni (non basato su etichette) per una migliore compatibilità con le convenzioni Kubernetes e altri strumenti.

Per informazioni dettagliate sulle fasi di sincronizzazione, gli hook e i pattern avanzati, consulta la documentazione di sincronizzazione con Argo CD.

Integrità dell'applicazione

Argo CD monitora lo stato di tutte le risorse dell'applicazione:

Stati di salute : * Integro: tutte le risorse funzionano come previsto * In corso di avanzamento - Risorse in fase di creazione o aggiornamento * Degradate - Alcune risorse non integre (pod bloccati, processi non riusciti) * Sospesa - Applicazione intenzionalmente sospesa * Mancante - Risorse definite in Git non presenti nel cluster

Argo CD include controlli di integrità integrati per le risorse Kubernetes più comuni (implementazioni, lavori, ecc.) e supporta controlli di integrità personalizzati per. StatefulSets CRDs

Lo stato dell'applicazione è determinato da tutte le sue risorse: se una risorsa lo è, l'applicazione sì. `Degraded` `Degraded`

Per ulteriori informazioni, consultate Resource Health nella documentazione del CD Argo.

Schemi multicluster

Argo CD supporta due modelli di implementazione principali:

H ub-and-spoke - Esegui Argo CD su un cluster di gestione dedicato che viene distribuito su più cluster di carichi di lavoro: * Controllo e visibilità centralizzati * Policy coerenti su tutti i cluster * Un'istanza Argo CD da gestire * Chiara separazione tra piano di controllo e carichi di lavoro

Per cluster: esegui Argo CD su ogni cluster, gestendo solo le applicazioni del cluster: * Separazione del cluster (un errore non influisce sugli altri) * Rete più semplice (nessuna comunicazione tra cluster) * Configurazione iniziale più semplice (nessuna registrazione del cluster)

Scegliete tra hub-and-spoke i team di piattaforma che gestiscono più cluster o per i team indipendenti per i cluster o quando i cluster devono essere completamente isolati.

Per una configurazione multi-cluster dettagliata, consulta. the section called “Considerazioni su Argo CD”

Progetti

I progetti forniscono il raggruppamento logico e il controllo degli accessi per le applicazioni:

- Restrizioni all'origine: limita i repository Git che possono essere utilizzati
- Restrizioni sulla destinazione: limita i cluster e i namespace a cui è possibile rivolgersi

- Restrizioni alle risorse: limita i tipi di risorse Kubernetes che possono essere distribuiti
- Integrazione RBAC: mappa i progetti all'utente e al gruppo di Identity Center AWS IDs

Le applicazioni appartengono a un singolo progetto. Se non specificato, usano il `default` progetto, che di default non ha restrizioni. Per uso in produzione, modificate il `default` progetto per limitare l'accesso e creare nuovi progetti con le restrizioni appropriate.

Per la configurazione del progetto e i modelli RBAC, vedere. the section called "Configurazione delle autorizzazioni "

Opzioni di sincronizzazione

Ottimizza il comportamento di sincronizzazione con opzioni comuni:

- `CreateNamespace=true`- Crea automaticamente lo spazio dei nomi di destinazione
- `ServerSideApply=true`- Utilizza server-side apply per una migliore risoluzione dei conflitti
- `SkipDryRunOnMissingResource=true`- Salta il dry run quando CRDs non esistono ancora (utile per le istanze kro)

```
spec:
  syncPolicy:
    syncOptions:
    - CreateNamespace=true
    - ServerSideApply=true
    - SkipDryRunOnMissingResource=true
```

Per un elenco completo delle opzioni di sincronizzazione, consultate la documentazione sulle opzioni di sincronizzazione di Argo CD.

## Fasi successive

- the section called "Configura i repository"- Configura l'accesso al repository Git
- the section called "Registra i cluster"- Registra i cluster di destinazione per la distribuzione
- the section called "Crea applicazioni"- Crea la tua prima applicazione
- the section called "Considerazioni su Argo CD"- Modelli specifici di EKS, integrazione con Identity Center e configurazione multicluster

- Documentazione su Argo CD - Documentazione completa su Argo CD, inclusi hook di sincronizzazione, controlli di integrità e modelli avanzati

## Configurare le autorizzazioni Argo CD

La funzionalità gestita da Argo CD si integra con AWS Identity Center per l'autenticazione e utilizza ruoli RBAC integrati per l'autorizzazione. Questo argomento spiega come configurare le autorizzazioni per utenti e team.

### Come funzionano le autorizzazioni con Argo CD

La funzionalità Argo CD utilizza AWS Identity Center per l'autenticazione e fornisce tre ruoli RBAC integrati per l'autorizzazione.

Quando un utente accede ad Argo CD:

1. Si autenticano utilizzando AWS Identity Center (che può essere federato con il tuo provider di identità aziendale)
2. AWS Identity Center fornisce informazioni su utenti e gruppi ad Argo CD
3. Argo CD associa utenti e gruppi ai ruoli RBAC in base alla configurazione
4. Gli utenti vedono solo le applicazioni e le risorse a cui hanno il permesso di accedere

### Built-in Ruoli RBAC

La funzionalità Argo CD offre tre ruoli integrati che puoi mappare agli utenti e ai gruppi di AWS Identity Center. Si tratta di ruoli con ambito  globale  che controllano l'accesso alle risorse di Argo CD come progetti, cluster e repository.

> **Important**
>
> I ruoli globali controllano l'accesso al CD Argo stesso, non alle risorse relative al progetto come le Applicazioni. Gli utenti di EDITOR e VIEWER non possono visualizzare o gestire le applicazioni per impostazione predefinita: hanno bisogno dei ruoli di progetto per accedere alle risorse relative al progetto. the section called "Ruoli del progetto e accesso nell'ambito del progetto"Per informazioni dettagliate sulla concessione dell'accesso alle applicazioni e ad altre risorse relative al progetto, consulta.

AMMINISTRATORE

Accesso completo a tutte le risorse e le impostazioni di Argo CD:

- Crea, aggiorna ed elimina applicazioni e ApplicationSets in qualsiasi progetto
- Gestisci la configurazione del CD Argo
- Registra e gestisci i cluster target di distribuzione
- Configurare l'accesso al repository
- Crea e gestisci progetti
- Visualizza lo stato e la cronologia di tutte le candidature
- Elenca e accedi a tutti i cluster e i repository

REDATTORE

Può aggiornare i progetti e configurare i ruoli del progetto, ma non può modificare le impostazioni globali di Argo CD:

- Aggiornare i progetti esistenti (non è possibile creare o eliminare progetti)
- Configura i ruoli e le autorizzazioni del progetto
- Visualizza le chiavi e i certificati GPG
- Impossibile modificare la configurazione globale di Argo CD
- Non è possibile gestire direttamente cluster o repository
- Impossibile visualizzare o gestire le applicazioni senza ruoli di progetto

VISUALIZZATORE

Read-only accesso alle risorse Argo CD:

- Visualizza le configurazioni del progetto
- Elenca tutti i progetti (inclusi i progetti a cui l'utente non è assegnato)
- Visualizza le chiavi e i certificati GPG
- Impossibile elencare cluster o repository
- Non è possibile apportare modifiche

- Impossibile visualizzare o gestire le applicazioni senza ruoli di progetto

> **Note**
>
> Per concedere agli utenti EDITOR o VIEWER l'accesso alle applicazioni, un ADMIN o EDITOR deve creare ruoli di progetto che associano i gruppi di Identity Center a autorizzazioni specifiche all'interno di un progetto.

## Ruoli del progetto e accesso nell'ambito del progetto

I ruoli globali (ADMIN, EDITOR, VIEWER) controllano l'accesso al CD Argo stesso. I ruoli del progetto controllano l'accesso alle risorse e alle funzionalità all'interno di un progetto specifico, tra cui:

- Risorse: applicazioni ApplicationSets, credenziali del repository, credenziali del cluster
- Funzionalità: accesso ai log, accesso esecutivo ai pod delle applicazioni

Comprensione del modello di autorizzazione a due livelli:

- Ambito globale: Built-in i ruoli determinano cosa possono fare gli utenti con progetti, cluster, repository e impostazioni Argo CD
- Ambito del progetto: i ruoli del progetto determinano cosa possono fare gli utenti con risorse e funzionalità all'interno di un progetto specifico

Ciò significa che:

- Gli utenti ADMIN possono accedere a tutte le risorse e funzionalità del progetto senza configurazioni aggiuntive
- Agli utenti EDITOR e VIEWER devono essere concessi i ruoli di progetto per accedere alle risorse e alle funzionalità del progetto
- Gli utenti EDITOR possono creare ruoli di progetto per concedere a se stessi e ad altri l'accesso all'interno dei progetti che possono aggiornare

Esempio di flusso di lavoro:

1. Un AMMINISTRATORE associa un gruppo di Identity Center al ruolo EDITOR a livello globale

2. Un ADMIN crea un progetto per un team
3. L'EDITOR configura i ruoli del progetto all'interno di quel progetto per concedere ai membri del team l'accesso alle risorse relative al progetto
4. I membri del team (che possono avere il ruolo globale di VIEWER) possono ora visualizzare e gestire le applicazioni in quel progetto in base alle autorizzazioni relative ai ruoli del progetto

Per i dettagli sulla configurazione dei ruoli del progetto, consulta. the section called "Project-based controllo degli accessi"

## Configurare le mappature dei ruoli

Associa gli utenti e i gruppi di AWS Identity Center ai ruoli di Argo CD durante la creazione o l'aggiornamento della funzionalità.

Esempio di mappatura dei ruoli:

```
{
  "rbacRoleMappings": {
    "ADMIN": ["AdminGroup", "alice@example.com"],
    "EDITOR": ["DeveloperGroup", "DevOpsTeam"],
    "VIEWER": ["ReadOnlyGroup", "bob@example.com"]
  }
}
```

**Note**

I nomi dei ruoli fanno distinzione tra maiuscole e minuscole e devono essere in maiuscolo (ADMIN, EDITOR, VIEWER).

**Important**

L'integrazione di EKS Capabilities con AWS Identity Center supporta fino a 1.000 identità per funzionalità Argo CD. Un'identità può essere un utente o un gruppo.

Aggiorna le mappature  dei ruoli:

```
aws eks update-capability \
  --region us-east-1 \
  --cluster-name cluster \
  --capability-name capname \
  --role-arn "arn:aws:iam::111122223333:role/EKSCapabilityRole" \
  --configuration '{
    "argoCd": {
      "rbacRoleMappings": {
        "addOrUpdateRoleMappings": [
          {
            "role": "ADMIN",
            "identities": [
              { "id": "686103e0-f051-7068-b225-e6392b959d9e", "type": "SSO_USER" }
            ]
          }
        ]
      }
    }
  }'
```

## Utilizzo dell'account amministratore

L'account amministratore è progettato per la configurazione iniziale e le attività amministrative come la registrazione dei cluster e la configurazione dei repository.

Quando l'account amministratore è appropriato:

- Configurazione e configurazione iniziali delle capacità
- Sviluppo individuale o dimostrazioni rapide
- Attività amministrative (registrazione del cluster, configurazione del repository, creazione del progetto)

Procedure consigliate per l'account amministratore:

- Non utilizzare i token dell'account per il controllo della versione
- Ruota immediatamente i token se esposti
- Limita l'utilizzo dei token dell'account alle attività di configurazione e amministrative
- Imposta tempi di scadenza brevi (massimo 12 ore)
- È possibile creare solo 5 token di account alla volta

Quando utilizzare invece l'accesso basato sul progetto:

- Ambienti di sviluppo condivisi con più utenti
- Qualsiasi ambiente che assomigli alla produzione
- Quando hai bisogno di percorsi di controllo su chi ha eseguito le azioni
- Quando è necessario applicare restrizioni relative alle risorse o limiti di accesso

Per ambienti di produzione e scenari multiutente, utilizza il controllo degli accessi basato su progetti con ruoli RBAC dedicati mappati ai gruppi di Identity Center. AWS

## Project-based controllo degli accessi

Utilizzate Argo CD Projects (AppProject) per fornire un controllo granulare degli accessi e l'isolamento delle risorse per i team.

> **Important**
>
> Prima di assegnare utenti o gruppi a ruoli specifici del progetto, è necessario innanzitutto associarli a un ruolo globale di Argo CD (ADMIN, EDITOR o VIEWER) nella configurazione delle funzionalità. Gli utenti non possono accedere ad Argo CD senza una mappatura globale dei ruoli, anche se sono assegnati ai ruoli del progetto.
> Prendi in considerazione la possibilità di mappare gli utenti al ruolo VIEWER a livello globale, quindi concedi autorizzazioni aggiuntive tramite ruoli specifici del progetto. Ciò fornisce l'accesso di base e consente al contempo un controllo dettagliato a livello di progetto.

I progetti forniscono:

- Restrizioni relative all'origine: limita i repository Git che possono essere utilizzati
- Restrizioni di destinazione: limita i cluster e i namespace che possono essere presi di mira
- Restrizioni sulle risorse: limita i tipi di risorse Kubernetes che possono essere distribuiti
- Integrazione RBAC: mappa i progetti ai gruppi AWS Identity Center o ai ruoli Argo CD

Esempio di progetto per l'isolamento del team:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
```

```
metadata:
  name: team-a
  namespace: argocd
spec:
  description: Team A applications

  # Required: Specify which namespaces this project watches for Applications
  sourceNamespaces:
  - argocd

  # Source restrictions
  sourceRepos:
  - https://github.com/myorg/team-a-apps

  # Destination restrictions
  destinations:
  - namespace: team-a-*
    server: arn:aws:eks:us-west-2:111122223333:cluster/production

  # Resource restrictions
  clusterResourceWhitelist:
  - group: ''
    kind: Namespace
  namespaceResourceWhitelist:
  - group: 'apps'
    kind: Deployment
  - group: ''
    kind: Service
  - group: ''
    kind: ConfigMap
```

### Namespace di origine

Quando si utilizza la funzionalità EKS Argo CD, il `spec.sourceNamespaces` campo è obbligatorio nelle definizioni. AppProject Questo campo specifica quale namespace può contenere applicazioni o ApplicationSets che fa riferimento a questo progetto.

> **Important**
>
> La funzionalità EKS Argo CD supporta solo un singolo namespace per le applicazioni e, in genere ApplicationSets, lo spazio dei nomi specificato durante la creazione della funzionalità. `argocd` Ciò è diverso dal CD Argo open source che supporta più namespace.

AppProject configurazione

Tutti AppProjects devono includere lo spazio dei nomi configurato della funzionalità in: `sourceNamespaces`

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a-project
  namespace: argocd
spec:
  description: Applications for Team A

  # Required: Specify the capability's configured namespace
 (configuration.argoCd.namespace)
  sourceNamespaces:
    - argocd  # Must match your capability's namespace configuration

  # Source repositories this project can deploy from
  sourceRepos:
    - 'https://github.com/my-org/team-a-*'

  # Destination restrictions
  destinations:
    - namespace: 'team-a-*'
      server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
```

> **Note**
>
> Se si omette lo spazio dei nomi della funzionalità da`sourceNamespaces`, Applications o ApplicationSets in tale namespace non possono fare riferimento a questo progetto, con conseguenti errori di distribuzione.

Assegna utenti ai progetti:

I ruoli del progetto garantiscono agli utenti di EDITOR e VIEWER l'accesso alle risorse del progetto (applicazioni ApplicationSets, credenziali del repository e del cluster) e alle funzionalità (log, exec). Senza i ruoli di progetto, questi utenti non possono accedere a queste risorse anche se dispongono di un accesso globale ai ruoli.

Gli utenti ADMIN hanno accesso a tutte le applicazioni senza bisogno di ruoli di progetto.

Esempio: concessione dell'accesso all'Applicazione ai membri del team

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  # ... project configuration ...

  sourceNamespaces:
  - argocd

  # Project roles grant Application-level access
  roles:
  - name: developer
    description: Team A developers - can manage Applications
    policies:
    - p, proj:team-a:developer, applications, *, team-a/*, allow
    - p, proj:team-a:developer, clusters, get, *, allow  # See cluster names in UI
    groups:
    - 686103e0-f051-7068-b225-e6392b959d9e  # Identity Center group ID

  - name: viewer
    description: Team A viewers - read-only Application access
    policies:
    - p, proj:team-a:viewer, applications, get, team-a/*, allow
    - p, proj:team-a:viewer, clusters, get, *, allow  # See cluster names in UI
    groups:
    - 786203e0-f051-7068-b225-e6392b959d9f  # Identity Center group ID
```

> **Note**
>
> `clusters, get, *, allow`Includi i ruoli del progetto per consentire agli utenti di visualizzare i nomi dei cluster nell'interfaccia utente. Senza questa autorizzazione, il cluster di destinazione viene visualizzato come «sconosciuto».

Comprendere le politiche relative ai ruoli del progetto:

Il formato della politica è: `p, proj:<project>:<role>, <resource>, <action>, <object>, <allow/deny>`

Politiche relative alle risorse:

- `applications, , team-a/, allow` - Accesso completo a tutte le applicazioni del progetto team-a
- `applications, get, team-a/*, allow` - Read-only accesso alle applicazioni
- `applications, sync, team-a/*, allow` - Può sincronizzare le applicazioni ma non create/delete
- `applications, delete, team-a/*, allow` - Può eliminare le applicazioni (usare con cautela)
- `applicationsets, , team-a/, allow` - Accesso completo a ApplicationSets
- `repositories, *, *, allow` - Accesso alle credenziali del repository
- `clusters, *, *, allow` - Accesso alle credenziali del cluster

Criteri di capacità:

- `logs, , team-a/, allow` - Accesso ai log delle applicazioni
- `exec, , team-a/, allow` - Accesso Exec ai pod delle applicazioni

> **Note**
>
> Gli utenti di EDITOR possono creare ruoli di progetto per concedere a se stessi e agli altri autorizzazioni all'interno dei progetti che possono aggiornare. Ciò consente ai responsabili del team di controllare l'accesso alle risorse relative al progetto per il proprio team senza richiedere l'intervento dell'AMMINISTRATORE.

> **Note**
>
> Utilizza gli ID di gruppo di Identity Center (non i nomi dei gruppi) nel campo. `groups` Puoi anche utilizzare gli ID utente di Identity Center per l'accesso di singoli utenti. Trova questi ID nella console di AWS Identity Center o utilizzando l'interfaccia a riga di AWS comando.

## Schemi di autorizzazione comuni

Schema 1: team amministrativo con accesso completo

```
{
  "rbacRoleMappings": {
    "ADMIN": ["PlatformTeam", "SRETeam"]
  }
}
```

Gli utenti ADMIN possono visualizzare e gestire tutte le risorse relative al progetto senza configurazioni aggiuntive.

Modello 2: i team leader gestiscono i progetti, gli sviluppatori accedono tramite i ruoli di progetto

```
{
  "rbacRoleMappings": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["TeamLeads"],
    "VIEWER": ["AllDevelopers"]
  }
}
```

1. ADMIN crea progetti per ogni team
2. I team leader (EDITOR) configurano i ruoli del progetto per garantire agli sviluppatori l'accesso alle risorse (applicazioni ApplicationSets, credenziali) e alle funzionalità (log, exec)
3. Gli sviluppatori (VIEWER) possono accedere solo alle risorse e alle funzionalità consentite dai loro ruoli di progetto

Schema 3: Team-based accesso con i ruoli del progetto

1. ADMIN crea progetti e assegna ai responsabili dei team il ruolo di EDITOR a livello globale
2. I team leader (EDITOR) assegnano ai membri del team i ruoli di progetto all'interno dei loro progetti
3. I membri del team hanno bisogno solo del ruolo globale di VIEWER: i ruoli del progetto forniscono l'accesso alle risorse e alle funzionalità del progetto

```
{
  "rbacRoleMappings": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["TeamLeads"],
```

```
    "VIEWER": ["AllDevelopers"]
  }
}
```

## Best practice

Usa i gruppi anziché i singoli utenti: associa i gruppi di AWS Identity Center ai ruoli Argo CD anziché ai singoli utenti per una gestione più semplice.

Inizia con il privilegio minimo: inizia con l'accesso a VIEWER e concedi EDITOR o ADMIN secondo necessità.

Usa i progetti per l'isolamento del team: crea progetti separati AppProjects per team o ambienti diversi per far rispettare i limiti.

Sfrutta la federazione di Identity Center: configura AWS Identity Center per la federazione con il tuo provider di identità aziendale per una gestione centralizzata degli utenti.

Revisioni periodiche degli accessi: rivedi periodicamente le mappature dei ruoli e le assegnazioni dei progetti per garantire livelli di accesso appropriati.

Limita l'accesso ai cluster: ricorda che Argo CD RBAC controlla l'accesso alle risorse e alle operazioni di Argo CD, ma non corrisponde a Kubernetes RBAC. Gli utenti con accesso ad Argo CD possono distribuire applicazioni nei cluster a cui Argo CD ha accesso. Limita i cluster a cui Argo CD può accedere e utilizza le restrizioni di destinazione del progetto per controllare dove possono essere distribuite le applicazioni.

## AWS autorizzazioni di servizio

Per utilizzare AWS i servizi direttamente nelle risorse dell'applicazione (senza creare risorse del repository), allega le autorizzazioni IAM richieste al Capability Role.

Grafici ECR per Helm:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    }
  ]
}
```

CodeCommit archivi:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections (GitHub GitLab, Bitbucket):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

Vedi the section called "Configura i repository" per i dettagli sull'utilizzo di queste integrazioni.

### Fasi successive

- the section called "Lavorare con Argo CD"- Scopri come creare applicazioni e gestire le distribuzioni
- the section called "Concetti di Argo CD"- Comprendi i concetti di Argo CD, inclusi i progetti
- the section called "Considerazioni sulle funzionalità EKS"- Rivedi le migliori pratiche di sicurezza per le funzionalità

## Lavorare con Argo CD

Con Argo CD, definisci le applicazioni nei repository Git e Argo CD le sincronizza automaticamente con i tuoi cluster Kubernetes. Ciò consente l'implementazione dichiarativa e controllata dalla versione delle applicazioni con rilevamento automatico della deriva.

### Prerequisiti

Prima di lavorare con Argo CD, è necessario:

- È stato creato un cluster EKS con la funzionalità Argo CD (vedi) the section called "Crea una funzionalità Argo CD"
- Un repository Git contenente Kubernetes si manifesta
- `kubectl`configurato per comunicare con il tuo cluster

### Attività comuni

I seguenti argomenti guidano l'utente nelle attività più comuni di Argo CD:

the section called "Configura i repository"- Configura Argo CD per accedere ai tuoi repository Git utilizzando AWS Secrets Manager o Kubernetes AWS CodeConnections Secrets.

the section called "Registra i cluster"- Registra i cluster di destinazione in cui Argo CD distribuirà le applicazioni.

the section called "Progetti"- Organizza le applicazioni e applica i limiti di sicurezza utilizzando Projects per ambienti multi-tenant.

the section called "Configurare le impostazioni"- Configura il comportamento di Argo CD utilizzando `argocd-cm` ConfigMap, inclusi i controlli di integrità personalizzati per le risorse personalizzate.

the section called "Crea applicazioni"- Crea applicazioni che vengono distribuite da repository Git con politiche di sincronizzazione automatiche o manuali.

the section called "ApplicationSets"- Utilizzabile ApplicationSets per distribuire applicazioni in più ambienti o cluster utilizzando modelli e generatori.

## Accedi all'interfaccia utente del CD di Argo

Accedi all'interfaccia utente del CD Argo tramite la console EKS:

1. Apri la console Amazon EKS
2. Seleziona il tuo cluster
3. Scegli la  scheda  Funzionalità
4. Scegli  Argo CD
5. Scegli  Open Argo CD UI

L'interfaccia utente fornisce la topologia visiva dell'applicazione, lo stato e la cronologia della sincronizzazione, lo stato e gli eventi delle risorse, i controlli di sincronizzazione manuale e la gestione delle applicazioni.

## URL dell'endpoint Argo CD

Quando crei una funzionalità Argo CD, ricevi un endpoint unico sia per l'interfaccia utente che per l'API di Argo CD. Il nome host dell'endpoint è  deterministico: utilizza valori che già conoscete, come il nome della capacità, il nome del cluster, l'ID dell'account e la regione. AWS AWS Se si elimina una funzionalità e la si ricrea con lo stesso nome del cluster e lo stesso nome della funzionalità, si ottiene lo stesso endpoint. Poiché il nome host è prevedibile, è possibile crearlo in anticipo. Ad esempio, puoi preconfigurare le integrazioni DNS, Automation o Git webhook prima che la funzionalità termini il provisioning. Per qualsiasi funzionalità, il `serverUrl` valore from `describe-capability` è l'endpoint definitivo.

> **Note**
>
> Poiché l'endpoint viene riutilizzato, le integrazioni che facevano riferimento alla funzionalità precedente (webhook Git, token e record DNS) si connettono automaticamente a quello nuovo, senza alcuna indicazione che l'istanza sia cambiata. Dopo aver ricreato una

funzionalità, verifica nuovamente tali integrazioni e ruota i segreti o i token dei webhook condivisi con l'istanza precedente.

### Recupera l'endpoint

Per ottenere l'endpoint esatto di una funzionalità, usa il `describe-capability` comando e leggi il valore. `serverUrl` Sostituisci *`region-code`* *`my-cluster`* e *`my-argocd`* con i tuoi valori:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.configuration.argoCd.serverUrl' \
  --output text
```

> **Note**
>
> Il `serverUrl` campo e i record DNS dell'endpoint sono disponibili dopo che lo stato della funzionalità è cambiato in. `ACTIVE`

### Formato dell'endpoint

L'endpoint Argo CD utilizza il seguente formato:

```
https://capability-name-hash-account-id.eks-capabilities.region-code.amazonaws.com
```

Il nome host è composto dai seguenti componenti:

- *`capability-name`*— Il nome della funzionalità Argo CD, normalizzato in minuscolo con ogni carattere di sottolineatura (_) sostituito da un trattino (). - Se il nome normalizzato è più lungo di 41 caratteri, viene troncato a 41 caratteri.
- *`hash`*— Un valore esadecimale di 8 caratteri che rende unico il nome host. Sono i primi 8 caratteri dell' SHA-256 hash del nome della funzionalità e del nome del cluster. I nomi sono uniti da una barra () *`capability-name/cluster-name`* . Usa i nomi esatti che hai specificato quando hai creato la funzionalità. Questo valore fornisce solo l'unicità e non è un controllo di sicurezza.
- *`account-id`*— Il tuo ID account a 12 cifre AWS .

- `eks-capabilities.region-code.amazonaws.com`— Il dominio del servizio EKS Capabilities per la tua AWS regione.

Ad esempio, una funzionalità `my-argocd` denominata `my-cluster` nel cluster nell'account `111122223333` nella `us-west-2` Regione ha un endpoint simile al seguente:

```
https://my-argocd-dc855fdf-111122223333.eks-capabilities.us-west-2.amazonaws.com
```

Il nome host utilizza il nome della funzionalità normalizzato, ma l'hash utilizza il nome grezzo specificato. Ad esempio, la funzionalità `Payments_GitOps` sul cluster viene `my-cluster` normalizzata `payments-gitops` nel nome host e il relativo hash proviene dalla stringa non elaborata: `Payments_GitOps/my-cluster`

```
https://payments-gitops-4ce0b382-111122223333.eks-capabilities.us-west-2.amazonaws.com
```

> **Note**
>
> Usa l'`describe-capability`output per confermare l'endpoint esatto per la tua regione. AWS

Puoi calcolare tu stesso il *hash* componente. Utilizzalo `sha256sum` su Linux o `shasum -a 256` su macOS. I comandi seguenti utilizzano la funzionalità di esempio `my-argocd` sul cluster`my-cluster`. Sostituisci i nomi esatti che hai specificato quando hai creato la funzionalità:

```
# Linux
printf '%s' "my-argocd/my-cluster" | sha256sum | cut -c1-8

# macOS
printf '%s' "my-argocd/my-cluster" | shasum -a 256 | cut -c1-8
```

Entrambi i comandi vengono `dc855fdf` restituiti.

Accesso e sicurezza agli endpoint

Il nome host dell'endpoint non è un segreto e saperlo non garantisce l'accesso all'istanza Argo CD. Il nome host espone le stesse informazioni non segrete che già appaiono nella funzionalità Amazon Resource Name (ARN). Le `list-capabilities` operazioni `create-capability`describe-

`capability`, e restituiscono questo ARN. AWS L'autenticazione Identity Center e il controllo degli accessi basato sui ruoli (RBAC) di Argo CD controllano l'accesso all'interfaccia utente e all'API di Argo CD.

Se configuri l'accesso privato all'endpoint, il nome host si risolve solo negli indirizzi IP privati associati all'endpoint VPC. Puoi accedere all'endpoint solo dal tuo VPC o dalle reti connesse. Per ulteriori informazioni, consulta the section called “Accesso privato agli endpoint”.

## Documentazione upstream

Per informazioni dettagliate sulle funzionalità di Argo CD:

- Documentazione Argo CD - Guida utente completa
- Specifiche dell'applicazione - Riferimento completo all'API dell'applicazione
- ApplicationSet Guida: ApplicationSet modelli ed esempi
- Argo CD GitHub - Codice sorgente ed esempi

## Configurare l'accesso al repository

Prima di distribuire le applicazioni, configura Argo CD per accedere ai tuoi repository Git e ai registri cartografici Helm. Argo CD supporta diversi metodi di autenticazione per GitHub, GitLab Bitbucket ed ECR. AWS CodeCommit AWS

> **Note**
>
> Per le integrazioni dirette dei AWS servizi (grafici ECR Helm, CodeCommit repository e CodeConnections), è possibile farvi riferimento direttamente nelle risorse dell'applicazione senza creare configurazioni di repository. Il Capability Role deve disporre delle autorizzazioni IAM richieste. Per informazioni dettagliate, vedi the section called “Configurazione delle autorizzazioni ”.

### Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD
- Archivi Git contenenti manifesti di Kubernetes
- `kubectl`configurato per comunicare con il tuo cluster

> **Note**
>
> AWS CodeConnections può connettersi a server Git situati nel AWS Cloud o in locale. Per ulteriori informazioni, consulta AWS CodeConnections.

Metodi di autenticazione

| Metodo | Caso d'uso | Autorizzazioni IAM richieste |
|---|---|---|
| Integrazione diretta con i servizi AWS | | |
| CodeCommit | Integrazione diretta con i AWS CodeCommit repository Git. Non è necessaria alcuna configurazione del repository. | `codecommit:GitPull` |
| CodeConnections | Connect o Bitbucket con autenticazione gestita. GitHub GitLab Richiede la configurazione della connessione. | `codeconnections:UseConnection` |
| Artefatti ECR OCI | Integrazione diretta con AWS ECR per grafici e immagini manifest OCI Helm. Non è necessaria alcuna configurazione del repository. | `arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly` |
| Configurazione del repository con credenziali | | |
| AWS Secrets Manager (nome utente/token) | Memorizza token o password di accesso personali. Abilita la rotazione delle credenziali senza accesso a Kubernetes. | `arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess` |
| AWS Secrets Manager (chiave SSH) | Usa l'autenticazione tramite chiave SSH. Abilita la rotazione delle credenziali senza accesso a Kubernetes. | `arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess` |

| Metodo | Caso d'uso | Autorizzazioni IAM richieste |
|---|---|---|
| AWS Secrets Manager (GitHub App) | GitHub Autenticazione dell'app con chiave privata. Abilita la rotazione delle credenziali senza accesso a Kubernetes. | `arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess` |
| Kubernetes Secret | Metodo Argo CD standard che utilizza segreti all'interno del cluster | Nessuna (autorizzazioni gestite da EKS Access Entry con Kubernetes RBAC) |

AWS Accesso diretto ai servizi

Per quanto riguarda AWS i servizi, è possibile farvi riferimento direttamente nelle risorse dell'applicazione senza creare configurazioni del repository. Il Capability Role deve disporre delle autorizzazioni IAM richieste.

CodeCommit repository

Fai riferimento ai CodeCommit repository direttamente nelle applicazioni:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  source:
    repoURL: https://git-codecommit.region.amazonaws.com/v1/repos/repository-name
    targetRevision: main
    path: kubernetes/manifests
```

Autorizzazioni necessarie per il ruolo di capacità:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codecommit:GitPull",
```

```
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections

Reference GitHub o GitLab tramite Bitbucket repository. CodeConnections Il formato dell'URL del repository è derivato dall'ARN della CodeConnections connessione.

Il formato dell'URL del repository è:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  source:
    repoURL: https://codeconnections.region.amazonaws.com/git-http/account-
id/region/connection-id/owner/repository.git
    targetRevision: main
    path: kubernetes/manifests
```

Autorizzazioni necessarie per il ruolo di capacità:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codeconnections:UseConnection",
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

Grafici ECR Helm

ECR memorizza i grafici Helm come artefatti OCI. Argo CD supporta due modi per farvi riferimento:

Formato Helm (consigliato per i grafici Helm):

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-helm
  namespace: argocd
spec:
  source:
    repoURL: account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: chart-version
    chart: chart-name
    helm:
      valueFiles:
        - values.yaml
```

Nota: non includere il `oci://` prefisso quando si utilizza il formato Helm. Usa il `chart` campo per specificare il nome del grafico.

Formato OCI (per artefatti OCI con manifesti Kubernetes):

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-oci
  namespace: argocd
spec:
  source:
    repoURL: oci://account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: artifact-version
    path: path-to-manifests
```

Nota: includi il prefisso quando usi il formato OCI. `oci://` Usa il `path` campo invece di. `chart`

Autorizzazioni Capability Role richieste: allega la policy gestita:

```
arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Questa politica include le autorizzazioni ECR necessarie:`ecr:GetAuthorizationToken`,, `ecr:BatchGetImage` e. `ecr:GetDownloadUrlForLayer`

### Utilizzo di AWS Secrets Manager

Archivia le credenziali del repository in Secrets Manager e fai riferimento ad esse nelle configurazioni di Argo CD Repository. L'utilizzo di Secrets Manager consente la rotazione automatica delle credenziali senza richiedere l'accesso RBAC a Kubernetes: le credenziali possono essere ruotate utilizzando le autorizzazioni IAM verso Secrets Manager e Argo CD legge automaticamente i valori aggiornati.

> **Note**
>
> Per il riutilizzo delle credenziali su più repository (ad esempio, tutti gli archivi di un'organizzazione), utilizza i modelli di credenziali di repository con. GitHub `argocd.argoproj.io/secret-type: repo-creds` Ciò offre un'esperienza utente migliore rispetto alla creazione di segreti di repository individuali. Per ulteriori informazioni, consultate Repository Credentials nella documentazione di Argo CD.

#### Autenticazione con nome utente e token

Per gli archivi HTTPS con token di accesso o password personali:

Crea il segreto in Secrets Manager:

```
aws secretsmanager create-secret \
  --name argocd/my-repo \
  --description "GitHub credentials for Argo CD" \
  --secret-string '{"username":"your-username","token":"your-personal-access-token"}'
```

Campi opzionali del certificato client TLS (per server Git privati):

```
aws secretsmanager create-secret \
  --name argocd/my-private-repo \
  --secret-string '{
    "username":"your-username",
    "token":"your-token",

 "tlsClientCertData":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCi4uLgotLS0tLUVORCBDRVJUSUZJQ0FURS0tL

 "tlsClientCertKey":"LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCi4uLgotLS0tLUVORCBQUklWQVRFIEtFWS0tLS
  }'
```

> **Note**
>
> I `tlsClientCertKey` valori `tlsClientCertData` and devono essere codificati in base 64.

Crea un Repository Secret facendo riferimento a Secrets Manager:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/my-repo-AbCdEf
  project: default
```

autenticazione con chiave SSH

Per l'accesso Git basato su SSH, memorizza la chiave privata come testo semplice (non JSON):

Crea il segreto con la chiave privata SSH:

```
aws secretsmanager create-secret \
  --name argocd/my-repo-ssh \
  --description "SSH key for Argo CD" \
  --secret-string "-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
...
-----END OPENSSH PRIVATE KEY-----"
```

Crea un repository segreto per SSH:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-ssh
  namespace: argocd
  labels:
```

```
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: git@github.com:your-org/your-repo.git
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/my-repo-ssh-
AbCdEf
  project: default
```

GitHub Autenticazione dell'app

Per GitHub l'autenticazione dell'app con una chiave privata:

Crea il segreto con le credenziali GitHub dell'app:

```
aws secretsmanager create-secret \
  --name argocd/github-app \
  --description "GitHub App credentials for Argo CD" \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
    "githubAppInstallationID":"12345678"
  }'
```

> **Note**
>
> Il `githubAppPrivateKeySecret` valore deve essere codificato in base64.

Campo opzionale per Enterprise: GitHub

```
aws secretsmanager create-secret \
  --name argocd/github-enterprise-app \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
    "githubAppInstallationID":"12345678",
    "githubAppEnterpriseBaseUrl":"https://github.example.com/api/v3"
  }'
```

Crea un repository segreto per GitHub l'app:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-github-app
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/github-app-
AbCdEf
  project: default
```

Modelli di credenziali di repository

Per il riutilizzo delle credenziali su più repository (ad esempio, tutti gli archivi di GitHub un'organizzazione o di un utente), utilizza i modelli di credenziali di repository con. `argocd.argoproj.io/secret-type: repo-creds` Ciò offre una migliore esperienza utente rispetto alla creazione di segreti di repository individuali per ogni repository.

Crea un modello di credenziali di repository:

```
apiVersion: v1
kind: Secret
metadata:
  name: github-org-creds
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repo-creds
stringData:
  type: git
  url: https://github.com/your-org
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/github-org-
AbCdEf
```

Questo modello di credenziali si applica a tutti gli archivi che corrispondono al prefisso URL. `https://github.com/your-org` È quindi possibile fare riferimento a qualsiasi repository di questa organizzazione in Applicazioni senza creare segreti aggiuntivi.

Per ulteriori informazioni, consultate Repository Credentials nella documentazione del CD Argo.

> **Important**
>
> Assicurati che al tuo IAM Capability Role sia `arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess` associata la policy gestita o autorizzazioni equivalenti, incluse `secretsmanager:GetSecretValue` le autorizzazioni di decrittografia KMS. Vedi per la configurazione delle policy IAMthe section called "Considerazioni su Argo CD".

Usando AWS CodeConnections

Per CodeConnections l'integrazione, vedithe section called " AWS CodeConnections".

CodeConnections fornisce l'autenticazione gestita per GitHub e Bitbucket senza memorizzare le credenziali. GitLab

Utilizzo di Kubernetes Secrets

Archivia le credenziali direttamente in Kubernetes utilizzando il metodo Argo CD standard.

Per HTTPS con token di accesso personale:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  username: your-username
  password: your-personal-access-token
```

Per SSH:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-ssh
  namespace: argocd
```

```
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: git@github.com:your-org/your-repo.git
  sshPrivateKey: |
    -----BEGIN OPENSSH PRIVATE KEY-----
    ... your private key ...
    -----END OPENSSH PRIVATE KEY-----
```

CodeCommit repository

Per AWS CodeCommit, concedi le CodeCommit autorizzazioni a IAM Capability Role ()`codecommit:GitPull`.

Configura il repository:

```
apiVersion: v1
kind: Secret
metadata:
  name: codecommit-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://git-codecommit.us-west-2.amazonaws.com/v1/repos/my-repo
  project: default
```

Per una configurazione dettagliata delle policy IAM, consultathe section called "Considerazioni su Argo CD".

Verifica la connessione al repository

Controlla lo stato della connessione tramite l'interfaccia utente di Argo CD in Impostazioni → Repository. L'interfaccia utente mostra lo stato della connessione e gli eventuali errori di autenticazione.

I Repository Secrets non includono informazioni sullo stato.

Risorse aggiuntive

- the section called "Registra i cluster"- Registra i cluster di destinazione per le distribuzioni

- the section called "Crea applicazioni"- Crea la tua prima applicazione
- the section called "Considerazioni su Argo CD"- Autorizzazioni IAM e configurazione di sicurezza
- Archivi privati: riferimento alla configurazione del repository upstream

## Registra i cluster di destinazione

Registra i cluster per consentire ad Argo CD di distribuire applicazioni su di essi. È possibile registrare lo stesso cluster su cui è in esecuzione Argo CD (cluster locale) o cluster remoti in account o regioni diversi. Una volta registrato, un cluster rimarrà in uno stato di connessione sconosciuto fino a quando non si crea un'applicazione all'interno di quel cluster. Per creare un'applicazione Argo CD dopo la registrazione del cluster, consultathe section called "Crea applicazioni".

### Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD
- `kubectl`configurato per comunicare con il cluster
- Per i cluster remoti: autorizzazioni IAM e voci di accesso appropriate

### Registra il cluster locale

Per distribuire le applicazioni nello stesso cluster su cui è in esecuzione Argo CD, registratelo come obiettivo di distribuzione.

**Important**

La funzionalità Argo CD non registra automaticamente il cluster locale. È necessario registrarlo in modo esplicito per distribuire le applicazioni nello stesso cluster. È possibile utilizzare il nome del cluster `in-cluster` per motivi di compatibilità con la maggior parte degli esempi di Argo CD online.

**Note**

Un EKS Access Entry viene creato automaticamente per il cluster locale con l'Argo CD Capability Role, ma per impostazione predefinita non vengono concesse autorizzazioni RBAC per Kubernetes. Ciò segue il principio del privilegio minimo: è necessario configurare in modo esplicito le autorizzazioni richieste da Argo CD in base al caso d'uso. Ad esempio, se si

utilizza questo cluster solo come hub Argo CD per gestire cluster remoti, non sono necessarie autorizzazioni di distribuzione locale. Per le opzioni di configurazione, consulta la sezione relativa ai requisiti RBAC di Access Entry di seguito.

Utilizzo dell'Argo CD CLI:

```
argocd cluster add <cluster-context-name> \
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/my-cluster \
  --name local-cluster
```

Utilizzo di un segreto Kubernetes:

```
apiVersion: v1
kind: Secret
metadata:
  name: local-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
stringData:
  name: local-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
  project: default
```

Applica la configurazione:

```
kubectl apply -f local-cluster.yaml
```

**Note**

Utilizza l'ARN del cluster EKS nel `server` campo, non l'URL del server API Kubernetes. La funzionalità gestita richiede che gli ARN identifichino i cluster. L'impostazione predefinita non `kubernetes.default.svc` è supportata.

Registra cluster remoti

Per eseguire la distribuzione su cluster remoti:

Fase 1: Creare la voce di accesso sul cluster remoto

Sostituisci *region-code* con la AWS regione in cui si trova il cluster remoto, sostituisci *remote-cluster* con il nome del cluster remoto e sostituisci l'ARN con il tuo ruolo di funzionalità Argo CD ARN.

```
aws eks create-access-entry \
  --region region-code \
  --cluster-name remote-cluster \
  --principal-arn arn:aws:iam::111122223333:role/ArgoCDCapabilityRole \
  --type STANDARD
```

Fase 2: Associare una politica di accesso alle autorizzazioni RBAC di Kubernetes

L'Access Entry richiede le autorizzazioni RBAC di Kubernetes per Argo CD per distribuire le applicazioni. Per iniziare rapidamente, puoi utilizzare: `AmazonEKSClusterAdminPolicy`

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name remote-cluster \
  --principal-arn arn:aws:iam::111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Fornisce l'accesso completo all'amministratore del cluster (equivalente `asystem:masters`). È utile per iniziare, ma non deve essere utilizzato in produzione. Per gli ambienti di produzione, utilizza autorizzazioni più restrittive associando l'Access Entry a gruppi Kubernetes personalizzati e creando ruoli o associazioni appropriati. ClusterRole Consulta la sezione sulla configurazione di produzione di seguito per la configurazione con privilegi minimi.

Fase 3: Registrare il cluster in Argo CD

Utilizzo dell'Argo CD CLI:

```
argocd cluster add <cluster-context-name> \
```

```
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/remote-cluster \
  --name remote-cluster
```

Utilizzo di un segreto Kubernetes:

```
apiVersion: v1
kind: Secret
metadata:
  name: remote-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
stringData:
  name: remote-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/remote-cluster
  project: default
```

Applica la configurazione:

```
kubectl apply -f remote-cluster.yaml
```

Cross-account cluster

Per eseguire la distribuzione su cluster con account diversi: AWS

1. Nell'account di destinazione, creare un Access Entry sul cluster EKS di destinazione utilizzando l'ARN Argo CD IAM Capability Role dall'account di origine come principale
2. Associa una politica di accesso alle autorizzazioni RBAC di Kubernetes appropriate
3. Registra il cluster in Argo CD utilizzando il relativo ARN del cluster EKS

Non è richiesta la creazione di ruoli IAM aggiuntivi o la configurazione delle policy di fiducia: EKS Access Entries gestisce l'accesso tra account.

Il formato ARN del cluster include la regione, quindi le distribuzioni interregionali utilizzano lo stesso processo delle distribuzioni nella stessa regione.

Verifica la registrazione del cluster

Visualizza i cluster registrati:

```
kubectl get secrets -n argocd -l argocd.argoproj.io/secret-type=cluster
```

Oppure controlla lo stato del cluster nell'interfaccia utente di Argo CD in Impostazioni → Cluster.

Cluster privati

La funzionalità Argo CD fornisce un accesso trasparente a cluster EKS completamente privati senza richiedere peering VPC o configurazioni di rete specializzate.

AWS gestisce automaticamente la connettività tra la funzionalità Argo CD e i cluster remoti privati.

È sufficiente registrare il cluster privato utilizzando la relativa ARN: non è richiesta alcuna configurazione di rete aggiuntiva.

Accedi ai requisiti RBAC di ingresso

Quando si crea una funzionalità Argo CD, viene creata automaticamente una voce EKS Access Entry per il ruolo Capability, ma per impostazione predefinita non vengono concesse autorizzazioni RBAC per Kubernetes. Questa progettazione intenzionale segue il principio del privilegio minimo: diversi casi d'uso richiedono autorizzazioni diverse.

Ad esempio: * Se si utilizza il cluster solo come hub Argo CD per gestire cluster remoti, non sono necessarie autorizzazioni di distribuzione locale* Se si distribuiscono applicazioni localmente, è necessario l'accesso in lettura a livello di cluster e l'accesso in scrittura a namespace specifici * Se è necessario creare CRD, sono necessarie autorizzazioni di amministratore del cluster aggiuntive

È necessario configurare in modo esplicito le autorizzazioni richieste da Argo CD in base alle proprie esigenze.

Autorizzazioni minime per Argo CD

Argo CD necessita di due tipi di autorizzazioni per funzionare senza errori:

Autorizzazioni di lettura (a livello di cluster): Argo CD deve essere in grado di leggere tutti i tipi di risorse e le Custom Resource Definitions (CRD) in tutto il cluster per:

- Scoperta delle risorse e controlli dello stato
- Rilevamento della deriva tra lo stato desiderato e quello effettivo
- Convalida delle risorse prima della distribuzione

Autorizzazioni di scrittura (specifiche dello spazio dei nomi): Argo CD richiede le autorizzazioni di creazione, aggiornamento ed eliminazione per le risorse definite in Applicazioni:

- Implementa i carichi di lavoro delle applicazioni (implementazioni, servizi, ecc.) ConfigMaps
- Applica risorse personalizzate (CRD specifiche per le tue applicazioni)
- Gestisci il ciclo di vita delle applicazioni

### Configurazione rapida

Per iniziare rapidamente, in ambienti di test o sviluppo, utilizza: `AmazonEKSClusterAdminPolicy`

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Fornisce l'accesso completo all'amministratore del cluster (equivalente `asystem:masters`), inclusa la possibilità di creare CRD, modificare le risorse a livello di cluster e distribuirle in qualsiasi spazio dei nomi. Questa funzionalità è utile per lo sviluppo e i POC, ma non deve essere utilizzata in produzione. Per la produzione, utilizzate la configurazione con privilegi minimi riportata di seguito.

### Configurazione di produzione con privilegi minimi

Per gli ambienti di produzione, crea un RBAC Kubernetes personalizzato che garantisca:

- Cluster-wide accesso in lettura a tutte le risorse (per il rilevamento e i controlli di integrità)
- Namespace-specific accesso in scrittura (per le distribuzioni)

#### Fase 1: Associare Access Entry a un gruppo Kubernetes personalizzato

```
aws eks associate-access-policy \
  --region region-code \
```

```
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy \
  --access-scope type=namespace,namespaces=app-namespace
```

Fase 2: ClusterRole Creazione per l'accesso in lettura

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: argocd-read-all
rules:
# Read access to all resources for discovery and health checks
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["get", "list", "watch"]
```

Fase 3: Creare un ruolo per l'accesso in scrittura ai namespace delle applicazioni

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: argocd-deploy
  namespace: app-namespace
rules:
# Full access to deploy application resources
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
```

Fase 4: Associare i ruoli al gruppo Kubernetes

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: argocd-read-all
subjects:
- kind: Group
  name: eks-access-entry:arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
  apiGroup: rbac.authorization.k8s.io
roleRef:
```

```
  kind: ClusterRole
  name: argocd-read-all
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: argocd-deploy
  namespace: app-namespace
subjects:
- kind: Group
  name: eks-access-entry:arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: argocd-deploy
  apiGroup: rbac.authorization.k8s.io
```

**Note**

Il formato del nome del gruppo per Access Entries è `eks-access-entry:` seguito dall'ARN principale. Ripetere l'operazione RoleBinding per ogni namespace in cui Argo CD deve distribuire le applicazioni.

**Important**

Argo CD deve essere in grado di leggere tutti i tipi di risorse del cluster a fini di verifica e individuazione dello stato, anche se viene distribuito solo su namespace specifici. Senza accesso in lettura a livello di cluster, Argo CD mostrerà errori durante il controllo dello stato delle applicazioni.

### Limita l'accesso ai cluster con Projects

Usa Projects per controllare in quali cluster e namespace le applicazioni possono essere distribuite configurando i cluster e i namespace di destinazione consentiti in: `spec.destinations`

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
```

```
metadata:
  name: production
  namespace: argocd
spec:
  destinations:
  - server: arn:aws:eks:us-west-2:111122223333:cluster/prod-cluster
    namespace: '*'
  - server: arn:aws:eks:eu-west-1:111122223333:cluster/prod-eu-cluster
    namespace: '*'
  sourceRepos:
  - 'https://github.com/example/production-apps'
```

Per informazioni dettagliate, vedi the section called "Progetti".

Risorse aggiuntive

- the section called "Progetti"- Organizza le applicazioni e applica i limiti di sicurezza
- the section called "Crea applicazioni"- Implementa la tua prima applicazione
- the section called "ApplicationSets"- Esegui la distribuzione su più cluster con ApplicationSets
- the section called "Considerazioni su Argo CD"- Multi-cluster modelli e configurazione tra account
- Configurazione dichiarativa del cluster - Riferimento alla configurazione del cluster upstream

## Lavorare con Argo CD Projects

Argo CD Projects (AppProject) forniscono il raggruppamento logico e il controllo degli accessi per le applicazioni. I progetti definiscono i repository Git, i cluster di destinazione e i namespace che le applicazioni possono utilizzare, abilitando limiti di multi-tenancy e sicurezza nelle istanze Argo CD condivise.

Quando usare i progetti

Usa Projects per:

- Separa le applicazioni per team, ambiente o unità aziendale
- Limita i repository da cui i team possono eseguire l'implementazione
- Limita i cluster e i namespace in cui i team possono effettuare l'implementazione
- Applica le quote di risorse e i tipi di risorse consentiti
- Fornisci l'implementazione self-service delle applicazioni con guardrail

### Progetto predefinito

Ogni funzionalità di Argo CD include un `default` progetto che consente l'accesso a tutti i repository, i cluster e i namespace. Sebbene sia utile per i test iniziali, crea progetti dedicati con restrizioni esplicite per l'uso in produzione.

Per i dettagli sulla configurazione predefinita del progetto e su come limitarla, consultate The Default Project nella documentazione di Argo CD.

### Crea un progetto

Crea un progetto applicando una `AppProject` risorsa al tuo cluster.

Esempio: Team-specific progetto

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  description: Applications for Team A

  # Source repositories this project can deploy from
  sourceRepos:
    - 'https://github.com/my-org/team-a-*'
    - 'https://github.com/my-org/shared-libs'


  # Source namespaces (required for EKS capability)
  sourceNamespaces:
    - argocd
    - team-a-dev
    - team-a-prod

  # Destination clusters and namespaces
  destinations:
    - name: dev-cluster
      namespace: team-a-dev
    - name: prod-cluster
      namespace: team-a-prod

  # Allowed resource types
```

```
  clusterResourceWhitelist:
    - group: ''
      kind: Namespace

  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
    - group: ''
      kind: ConfigMap
```

Applica il progetto:

```
kubectl apply -f team-a-project.yaml
```

Configurazione del progetto

Repository di origine

Controlla quali repository Git le applicazioni di questo progetto possono utilizzare:

```
spec:
  sourceRepos:
    - 'https://github.com/my-org/app-*'  # Wildcard pattern
    - 'https://github.com/my-org/infra'  # Specific repo
```

È possibile utilizzare caratteri jolly e schemi di negazione (`!`prefisso) per consentire o negare repository specifici. Per i dettagli, consultate Gestione dei progetti nella documentazione di Argo CD.

Namespace di origine

Quando si utilizza la funzionalità EKS Argo CD, il `spec.sourceNamespaces` campo è obbligatorio nella definizione personalizzata. AppProject Questo campo specifica quali namespace possono contenere applicazioni o ApplicationSets che fanno riferimento a questo progetto:

> **Important**
>
> Questo è un campo obbligatorio per la funzionalità EKS Argo CD, che differisce da OSS Argo CD dove questo campo è facoltativo.

Comportamento predefinito AppProject

Include `default` AppProject automaticamente lo spazio dei `argocd` nomi in. `sourceNamespaces` Se devi creare applicazioni o ApplicationSets in namespace aggiuntivi, modifica il `sourceNamespaces` campo per aggiungere quei namespace:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: default
  namespace: argocd
spec:
  sourceNamespaces:
    - argocd           # Already included by default
    - team-a-apps      # Add additional namespaces as needed
    - team-b-apps
```

Configurazione personalizzata AppProject

Quando si crea una personalizzazione AppProject, è necessario includere manualmente lo spazio dei nomi di `argocd` sistema e qualsiasi altro spazio dei nomi in cui si intende creare applicazioni oppure: ApplicationSets

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a-project
  namespace: argocd
spec:
  description: Applications for Team A

  # Required: Manually specify all namespaces
  sourceNamespaces:
    - argocd           # ArgoCD system namespace (required)
    - team-a-dev       # Custom namespace for dev Applications
    - team-a-prod      # Custom namespace for prod Applications

  # Source repositories this project can deploy from
  sourceRepos:
    - 'https://github.com/my-org/team-a-*'

  # Destination restrictions
```

```
  destinations:
    - namespace: 'team-a-*'
      server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster  # Use cluster ARN
from: aws eks describe-cluster
```

> **Note**
>
> Se omettete uno spazio dei nomi `dasourceNamespaces`, Applications o ApplicationSets create in quel namespace non saranno in grado di fare riferimento a questo progetto, con conseguenti errori di distribuzione.

Restrizioni sulla destinazione

Limite in cui le applicazioni possono essere distribuite:

```
spec:
  destinations:
    - name: prod-cluster  # Specific cluster by name
      namespace: production
    - name: '*'  # Any cluster
      namespace: team-a-*  # Namespace pattern
```

> **Important**
>
> Utilizza nomi di cluster e modelli di namespace specifici anziché caratteri jolly per i progetti di produzione. In questo modo si evitano implementazioni accidentali in cluster o namespace non autorizzati.

È possibile utilizzare caratteri jolly e schemi di negazione per controllare le destinazioni. Per i dettagli, consultate Gestione dei progetti nella documentazione del CD Argo.

Restrizioni sulle risorse

Controlla quali tipi di risorse Kubernetes possono essere implementati:

Cluster-scoped risorse:

```
spec:
```

```
  clusterResourceWhitelist:
    - group: ''
      kind: Namespace
    - group: 'rbac.authorization.k8s.io'
      kind: Role
```

Namespace-scoped risorse:

```
spec:
  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
    - group: ''
      kind: ConfigMap
    - group: 's3.services.k8s.aws'
      kind: Bucket
```

Usa le liste nere per negare risorse specifiche:

```
spec:
  namespaceResourceBlacklist:
    - group: ''
      kind: Secret  # Prevent direct Secret creation
```

Assegna applicazioni ai progetti

Quando crei un'applicazione, specifica il progetto nel `spec.project` campo:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  project: team-a  # Assign to team-a project
  source:
    repoURL: https://github.com/my-org/my-app
    path: manifests
  destination:
    name: prod-cluster
```

```
    namespace: team-a-prod
```

Le applicazioni senza un progetto specificato utilizzano il `default` progetto.

Ruoli del progetto e RBAC

I progetti possono definire ruoli personalizzati per un controllo granulare degli accessi. Associa i ruoli del progetto agli utenti e ai gruppi di AWS Identity Center nella configurazione delle funzionalità per controllare chi può sincronizzare, aggiornare o eliminare le applicazioni.

Esempio: progetto con ruoli di sviluppatore e amministratore

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  sourceRepos:
    - '*'
  destinations:
    - name: '*'
      namespace: 'team-a-*'

  roles:
    - name: developer
      description: Developers can sync applications
      policies:
        - p, proj:team-a:developer, applications, sync, team-a/*, allow
        - p, proj:team-a:developer, applications, get, team-a/*, allow
      groups:
        - team-a-developers

    - name: admin
      description: Admins have full access
      policies:
        - p, proj:team-a:admin, applications, *, team-a/*, allow
      groups:
        - team-a-admins
```

Per i dettagli sui ruoli del progetto, sui token JWT per le CI/CD pipeline e sulla configurazione RBAC, consulta Project Roles nella documentazione di Argo CD.

## Schemi comuni

### Environment-based Progetti

Crea progetti separati per ogni ambiente:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: production
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/*'
  destinations:
    - name: prod-cluster
      namespace: '*'
  # Strict resource controls for production
  clusterResourceWhitelist: []
  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
```

### Team-based Progetti

Isolare i team con progetti dedicati:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: platform-team
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/platform-*'
  destinations:
    - name: '*'
      namespace: 'platform-*'
  # Platform team can manage cluster resources
  clusterResourceWhitelist:
    - group: '*'
```

```
      kind: '*'
```

Multi-cluster Progetti

Implementa su più cluster con politiche coerenti:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: global-app
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/global-app'
  destinations:
    - name: us-west-cluster
      namespace: app
    - name: eu-west-cluster
      namespace: app
    - name: ap-south-cluster
      namespace: app
```

Best practice

Inizia con progetti restrittivi: inizia con autorizzazioni limitate ed espandi secondo necessità anziché iniziare con un accesso ampio.

Usa modelli di namespace: sfrutta i caratteri jolly nelle restrizioni dello spazio dei nomi (ad esempio`team-a-*`) per consentire flessibilità pur mantenendo i limiti.

Progetti di produzione separati: utilizza progetti dedicati per la produzione con controlli più rigorosi e politiche di sincronizzazione manuale.

Scopi del progetto di documentazione: utilizza il `description` campo per spiegare a cosa serve ogni progetto e chi dovrebbe utilizzarlo.

Rivedi regolarmente le autorizzazioni del progetto: controlla periodicamente i progetti per assicurarti che le restrizioni siano ancora in linea con le esigenze del team e i requisiti di sicurezza.

Risorse aggiuntive

- the section called “Configurazione delle autorizzazioni ”- Configurazione dell'integrazione tra RBAC e Identity Center

- the section called "Crea applicazioni"- Crea applicazioni all'interno dei progetti
- the section called "ApplicationSets"- Utilizzabile ApplicationSets con Projects per implementazioni multi-cluster
- Documentazione sui progetti Argo CD - Riferimento completo a upstream

## Configurare le impostazioni del CD Argo

Il CD EKS Capability for Argo offre un'esperienza Argo CD completamente gestita. Upstream Argo CD offre molte impostazioni e funzionalità opzionali e la funzionalità ne supporta un sottoinsieme. Per le impostazioni supportate, è possibile configurarle allo stesso modo del CD Argo upstream, tramite l'interfaccia del cluster. `argocd-cm` ConfigMap La funzionalità legge i campi supportati ConfigMap e li applica all'istanza Argo CD gestita.

Le sezioni seguenti descrivono come configurare `argocd-cm` ConfigMap le impostazioni supportate.

### Prerequisiti

Prima di configurare le impostazioni del CD Argo, è necessario disporre di:

- È stato creato un cluster EKS con la funzionalità Argo CD (vedi) the section called "Crea una funzionalità Argo CD"
- Lo spazio dei nomi configurato per Argo CD nella funzionalità (per impostazione predefinita, lo spazio dei nomi) `argocd`
- La `kubectl` CLI configurata per comunicare con il cluster

### Configura argocd-cm ConfigMap

Per configurare le impostazioni del CD Argo supportate, create un ConfigMap nome `argocd-cm` nel cluster. La funzionalità gestita legge le impostazioni supportate da questo ConfigMap e le applica all'istanza Argo CD gestita. Per le impostazioni supportate dalla funzionalità e per come vengono applicate, vederethe section called "Impostazioni supportate".

Creane uno ConfigMap con i seguenti requisiti:

- Assegna un nome al ConfigMap `argocd-cm`.
- Createlo nello spazio dei nomi configurato per Argo CD nella funzionalità (lo spazio dei nomi impostato nella configurazione di Argo CD quando avete creato la funzionalità). Per impostazione predefinita, questo è il namespace. `argocd`

- Applica l'etichetta. `app.kubernetes.io/part-of: argocd` Questa etichetta è obbligatoria e corrisponde al comportamento originario di Argo CD.
- Usa lo stesso formato di campo e le stesse chiavi del CD Argo originale.

L'esempio seguente mostra la ConfigMap struttura, con un'impostazione che mostra un banner nell'interfaccia utente di Argo CD. Aggiungi altre impostazioni supportate `data` in allo stesso modo.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-cm
  namespace: argocd
  labels:
    app.kubernetes.io/part-of: argocd
data:
  ui.bannercontent: "Production cluster"
```

**Important**

A non ConfigMap è un archivio sicuro. Non inserire segreti, credenziali o altre informazioni sensibili nel `argocd-cm` ConfigMap.

In che modo la funzionalità applica le tue impostazioni

È possibile configurare Argo CD creandone uno `argocd-cm` ConfigMap interno al proprio cluster. La funzionalità applica le impostazioni supportate dall'istanza Argo CD gestita dall'utente ConfigMap a quella gestita. Applica solo le impostazioni supportate e ignora qualsiasi altro campo o funzionalità impostato. Qualsiasi impostazione non elencata in non the section called “Impostazioni supportate” è supportata e non ha alcun effetto.

La funzionalità convalida i valori impostati. Se un valore non è valido o non è valido, la funzionalità lo ignora e continua a funzionare con la configurazione predefinita per tale impostazione. Un errore nella tua istanza Argo CD gestita ConfigMap non interrompe.

**Note**

La funzionalità applica la configurazione presente `argocd-cm` ConfigMap nel cluster. Qualsiasi principale con accesso in scrittura ConfigMap può modificare la configurazione

dell'istanza Argo CD gestita. Il controllo degli accessi basato sui ruoli (RBAC) di Kubernetes del tuo cluster regola l'accesso alle autorizzazioni IAM che controllano la risorsa di capacità ConfigMap, e non le autorizzazioni IAM. Come best practice di sicurezza, concedi l'autorizzazione a modificare gli oggetti nel namespace Argo CD solo a utenti e account di servizio affidabili. Poiché Kubernetes RBAC classifica le autorizzazioni in base al tipo di risorsa, puoi comunque concedere l'accesso di cui hanno bisogno gli altri utenti. Ad esempio, puoi consentire agli sviluppatori di creare e gestire le applicazioni ma non di modificarle. ConfigMaps Ciò impedisce loro di modificare la `argocd-cm` configurazione.
Per ulteriori informazioni sul modello di responsabilità condivisa, sull'RBAC di Kubernetes e sull'isolamento dello spazio dei nomi per la funzionalità Argo CD, vedere. the section called "Considerazioni sulle funzionalità EKS" Per controllare l'accesso all'interno di Argo CD, vedere. the section called "Configurazione delle autorizzazioni "

Impostazioni supportate

Le sezioni seguenti elencano le `argocd-cm` impostazioni supportate dalla funzionalità gestita, raggruppate per categoria. Ogni impostazione utilizza la stessa chiave e lo stesso formato del CD Argo originale. La colonna How your value is apply di ogni tabella mostra se il valore si aggiunge o sovrascrive la configurazione predefinita della funzionalità. Per la descrizione completa di ciascuna impostazione, consultate il ConfigMap riferimento argocd-cm sul sito Web della documentazione del CD di Argo.

Interfaccia utente

Queste impostazioni personalizzano l'interfaccia utente di Argo CD.

| Impostazione | Description | Come viene applicato il tuo valore |
|---|---|---|
| `ui.bannercontent` | Testo per un banner visualizzato nell'interfaccia utente, ad esempio un identificatore di ambiente o un avviso di manutenzione. | Sostituisce |
| `ui.bannerurl` | URL a cui si collega il banner, ad esempio un runbook o una pagina wiki. | Sostituisce |

| Impostazione | Description | Come viene applicato il tuo valore |
|---|---|---|
| `ui.bannerpermanent` | Impostato per impedire `true` agli utenti di chiudere il banner. | Sostituisce |
| `ui.bannerposition` | Dove appare il banner:`top`,`bottom`, o. `both` | Sostituisce |
| `ui.cssurl` | URL di un file CSS personalizzato per il branding o lo stile. Il CSS viene eseguito nel tuo browser. | Sostituisce |

Impostazioni delle risorse

Queste impostazioni controllano il modo in cui la funzionalità controlla, confronta e visualizza le risorse gestite da Argo CD.

| Impostazione | Description | Come viene applicato il tuo valore |
|---|---|---|
| `resource.customizations.ignoreDifferences.<group>_<kind>` | Campi da ignorare quando Argo CD confronta Git con il cluster per un tipo di risorsa, come il conteggio delle repliche gestito da un Horizontal Pod Autoscaler. | Aggiunge |
| `resource.customizations.ignoreDifferences.all` | Campi da ignorare quando si confronta Git con il cluster, applicati a tutti i tipi di risorse. | Aggiunge |
| `resource.customizations.ignoreResourceUpdates.<group>_<kind>` | Campi che Argo CD ignora quando decide se un evento di aggiornamento deve attivare la riconciliazione, con conseguente riduzione del carico. | Aggiunge |

| Impostazione | Description | Come viene applicato il tuo valore |
|---|---|---|
| | L'evento si verifica ancora e Argo CD ignora solo le modifiche a questi campi. | |
| `resource.customizations.ignoreResourceUpdates.all` | Campi che Argo CD ignora durante l'elaborazione degli eventi di aggiornamento, applicati a tutti i tipi di risorse. | Aggiunge |
| `resource.customizations.knownTypeFields.<group>_<kind>` | Tipi di campo (elenco, mappa o primitiva) per una risorsa personalizzata, in modo che Argo CD calcoli le differenze accurate invece di mostrare la sostituzione dell'intero campo. | Aggiunge |
| `resource.customizations.health.<group>_<kind>` | Controllo dello stato personalizzato per un tipo di risorsa, definito come script Lua. La funzionalità include controlli di integrità integrati per le risorse ACK e kro. Consulta the section called "Controlli dell'integrità personalizzati". | Sostituisce |
| `resource.exclusions` | Tipi di risorse che Argo CD non controlla, il che migliora le prestazioni per i tipi ad alto tasso di abbandono. | Aggiunge |
| `resource.inclusions` | Tipi di risorse che Argo CD controlla. Una volta impostato, Argo CD guarda solo i tipi elencati. | Aggiunge |
| `resource.compareoptions` | Opzioni che controllano il modo in cui Argo CD calcola le differenze, ad esempio. `ignoreAggregatedRoles` | Sostituisce |

| Impostazione | Description | Come viene applicato il tuo valore |
|---|---|---|
| `resource.respectRBAC` | Se il controller controlla solo le risorse per le quali dispone dell'autorizzazione RBAC per leggere. Accetta `normal` o. `strict` | Sostituisce |
| `resource.customLabels` | Etichette di risorse aggiuntive da visualizzare nella vista delle risorse dell'interfaccia utente. | Sostituisce |
| `resource.includeEventLabelKeys` | Etichette da applicazioni e progetti da copiare negli eventi Kubernetes generati da Argo CD. | Sostituisce |
| `resource.excludeEventLabelKeys` | Etichette da escludere dagli eventi Kubernetes generati da Argo CD. | Sostituisce |
| `resource.sensitive.mask.annotations` | Annotazioni da mascherare quando l'interfaccia utente o la CLI visualizzano i segreti. | Sostituisce |

Impostazioni del repository e degli strumenti

Queste impostazioni controllano gli strumenti dei manifest utilizzati da Argo CD per renderizzare i manifesti.

| Impostazione | Description | Come viene applicato il tuo valore |
|---|---|---|
| `kustomize.enable` | Se Kustomize è abilitato come tipo di sorgente manifesto. | Sostituisce |
| `helm.enable` | Se Helm è abilitato come tipo di sorgente manifesto. | Sostituisce |

| Impostazione | Description | Come viene applicato il tuo valore |
|---|---|---|
| `jsonnet.enable` | Se Jsonnet è abilitato come tipo di origine manifesto. | Sostituisce |
| `kustomize.buildOptions` | I flag globali della riga di comando sono passati a tutti. `kustomize build` La funzionalità supporta un sottoinsieme di flag. Consulta the section called "Flag Kustomize.buildOptions supportati". | Sostituisce |

Flag Kustomize.buildOptions supportati

Per`kustomize.buildOptions`, la funzionalità filtra il tuo valore in una serie di flag sicuri e supportati. Non supporta i flag che consentono alla build di leggere file arbitrari o eseguire codice arbitrario. Elimina ogni flag non supportato o valore non valido singolarmente e applica i flag supportati rimanenti. È possibile scrivere i flag nel modulo or. `--flag value` `--flag=value`

| Flag | Valori supportati | Note |
|---|---|---|
| `--reorder` | `legacy`, `none` | Cambia l'ordine solo dello YAML renderizzato. |
| `--enable-helm` | Booleano | Esegue il binario Helm gestito dal percorso. |
| `--enable-managedby-label` | Booleano | Aggiunge solo etichette. |

La funzionalità elimina qualsiasi altro contrassegno`--load-restrictor`, incluso`--enable-exec`, e`--enable-alpha-plugins`.

Controlli dell'integrità personalizzati

Argo CD valuta lo stato delle risorse che impiega. Per le risorse Kubernetes standard come Deployments and Services, Argo CD dispone di una logica di integrità integrata. Per le risorse

personalizzate che Argo CD non riconosce, non ha una logica di integrità integrata e non riporta lo stato di integrità.

Quando una risorsa personalizzata non viene sottoposta a controlli sullo stato di salute, Argo CD non ne segnala lo stato e la esclude dallo stato generale dell'applicazione. Di conseguenza, un'applicazione può segnalare `Healthy` anche quando le sue risorse sono ancora in fase di provisioning o hanno fallito. Ciò significa anche che le ondate di sincronizzazione possono avanzare prima che tali risorse siano pronte, poiché l'ordine di sincronizzazione dipende dallo stato di salute segnalato.

Con i controlli di integrità personalizzati, puoi definire la logica di integrità per le tue risorse personalizzate, in modo che Argo CD registri lo stato di salute accurato e sequenzi correttamente le implementazioni. Definite i controlli di integrità personalizzati nello stesso modo in cui utilizzate il CD Argo originale, utilizzando le stesse chiavi di configurazione. Gli script upstream esistenti e gli esempi della community funzionano con il CD EKS Capability for Argo senza modifiche.

Built-in controlli sanitari per ACK e kro

Il CD EKS Capability for Argo include controlli di integrità integrati per le risorse AWS Controllers for Kubernetes (ACK) e kro (Kube Resource Orchestrator). Queste risorse riportano uno stato di salute accurato senza configurazioni aggiuntive.

Per modificare il modo in cui la funzionalità valuta lo stato di una risorsa ACK o kro, puoi definire un controllo di integrità personalizzato per quel tipo di risorsa. Un controllo di integrità personalizzato definito per un tipo di risorsa sostituisce il controllo di integrità integrato per quel tipo.

Scrivi un controllo sanitario personalizzato

Definisci un controllo di integrità personalizzato aggiungendo uno script Lua a `argocd-cm` ConfigMap, utilizzando una chiave nel seguente formato:

```
resource.customizations.health.<group>_<kind>
```

Sostituiscilo *`<group>`* con il gruppo API della Custom Resource e *`<kind>`* con il suo genere. Ad esempio, la chiave per una risorsa personalizzata con il gruppo API `example.com` e il tipo `Database` è`resource.customizations.health.example.com_Database`.

Lo script Lua ha accesso all'oggetto risorsa tramite la `obj` variabile globale. Lo script deve restituire una tabella con un `status` campo impostato su uno tra`Healthy`, `ProgressingDegraded`,

oSuspended. Lo script può anche impostare un `message` campo opzionale per fornire un messaggio di stato descrittivo.

L'esempio seguente ConfigMap definisce un controllo dello stato di salute per una risorsa `Database` personalizzata. Lo script riporta la risorsa nello stato in `Healthy` cui si trova `Ready` e come `Progressing` diversamente:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-cm
  namespace: argocd
  labels:
    app.kubernetes.io/part-of: argocd
data:
  resource.customizations.health.example.com_Database: |
    hs = {}
    hs.status = "Progressing"
    hs.message = "Waiting for the resource to become ready"
    if obj.status ~= nil then
      if obj.status.phase == "Ready" then
        hs.status = "Healthy"
        hs.message = "Database is ready"
      end
    end
    return hs
```

Per ulteriori informazioni sul formato dello script di controllo dello stato, sull'elenco dei controlli di integrità integrati e sugli esempi della community che è possibile adattare, consultate Resource Health sul sito Web della documentazione del CD di Argo.

Sicurezza e limitazioni

Grazie alla funzionalità gestita, gli script di controllo dello stato personalizzati vengono eseguiti in un sistema di calcolo isolato e completamente gestito. L'ambiente di esecuzione è isolato per funzionalità e non ha accesso ai dati del cluster o alle AWS API. Non esegui il provisioning, non applichi patch o gestisci alcuna parte dell'ambiente di esecuzione.

Tieni presente quanto segue quando scrivi controlli di integrità personalizzati da utilizzare con EKS Capability:

- Le librerie Lua standard non sono disponibili. L'`useOpenLibs`opzione è sempre disabilitata, che è l'impostazione predefinita nel CD Argo upstream. Gli script non possono accedere al sistema operativo o al file system. Se si esegue la migrazione di uno script da un CD Argo autogestito che si basa su librerie Lua standard, potrebbe non funzionare allo stesso modo nella funzionalità. Si consiglia di testare gli script di controllo dello stato di salute in un ambiente di sviluppo prima di utilizzarli in produzione.

Se la valutazione dello stato di integrità è temporaneamente non disponibile, i report di funzionalità hanno interessato le Risorse personalizzate `Progressing` anziché rimuoverne lo stato di integrità. Ciò mantiene le risorse interessate visibili nello stato dell'Applicazione fino al ripristino della valutazione.

Verifica un controllo sanitario personalizzato

Dopo aver applicato o aggiornato il `argocd-cm` ConfigMap, conferma che il controllo sanitario è attivo:

1. Nell'interfaccia utente di Argo CD, scegliete un'applicazione che includa una risorsa personalizzata del tipo per il quale avete definito un controllo di integrità. Conferma che la risorsa riporti lo stato di integrità restituito dallo script. In alternativa, `argocd app get` *`<application-name>`* esegui e rivedi lo stato di integrità della risorsa.
2. Se la risorsa non riporta lo stato di salute previsto, verifica quanto segue:
   - ConfigMap È denominato `argocd-cm` e si trova nello spazio dei nomi configurato per Argo CD nella funzionalità.
   - ConfigMap Ha l'etichetta richiesta. `app.kubernetes.io/part-of: argocd`
   - La chiave di controllo dello stato utilizza la chiave corretta `<group>_<kind>` per il tipo di risorsa.

## Creazione di applicazioni

Le applicazioni rappresentano distribuzioni nei cluster di destinazione. Ogni applicazione definisce una fonte (repository Git) e una destinazione (cluster e namespace). Una volta applicato, Argo CD creerà le risorse specificate dai manifest nel repository Git nello spazio dei nomi del cluster. Le applicazioni spesso specificano le distribuzioni dei carichi di lavoro, ma possono gestire qualsiasi risorsa Kubernetes disponibile nel cluster di destinazione.

### Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD
- Accesso al repository configurato (vedi) the section called "Configura i repository"
- Cluster di destinazione registrato (vedithe section called "Registra i cluster")
- `kubectl`configurato per comunicare con il cluster

### Crea un'applicazione di base

Definisci un'applicazione che viene distribuita da un repository Git:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps
    targetRevision: HEAD
    path: guestbook
  destination:
    name: in-cluster
    namespace: default
```

> **Note**
>
> Usalo `destination.name` con il nome del cluster che hai usato durante la registrazione del cluster (come `in-cluster` per il cluster locale). Il `destination.server` campo funziona anche con gli ARN dei cluster EKS, ma si consiglia di utilizzare i nomi dei cluster per una migliore leggibilità.

Applica l'applicazione:

```
kubectl apply -f application.yaml
```

Visualizza lo stato dell'applicazione:

```
kubectl get application guestbook -n argocd
```

Configurazione dell'origine

Archivio Git:

```
spec:
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: main
    path: kubernetes/manifests
```

Tag o commit Git specifici:

```
spec:
  source:
    targetRevision: v1.2.0  # or commit SHA
```

Diagramma di Helm:

```
spec:
  source:
    repoURL: https://github.com/example/helm-charts
    targetRevision: main
    path: charts/my-app
    helm:
      valueFiles:
      - values.yaml
      parameters:
      - name: image.tag
        value: v1.2.0
```

Grafico Helm con valori dal repository Git esterno (modello multi-source):

```
spec:
  sources:
  - repoURL: https://github.com/example/helm-charts
    targetRevision: main
    path: charts/my-app
    helm:
```

```
      valueFiles:
      - $values/environments/production/values.yaml
  - repoURL: https://github.com/example/config-repo
    targetRevision: main
    ref: values
```

Per ulteriori informazioni, consultate Helm Value Files from External Git Repository nella documentazione del CD Argo.

Diagramma Helm tratto da ECR:

```
spec:
  source:
    repoURL: oci://account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: chart-version
    chart: chart-name
```

Se il Capability Role dispone delle autorizzazioni ECR richieste, il repository viene utilizzato direttamente e non è richiesta alcuna configurazione del repository. Per informazioni dettagliate, vedi the section called "Configura i repository".

Repository Git da CodeCommit:

```
spec:
  source:
    repoURL: https://git-codecommit.region.amazonaws.com/v1/repos/repository-name
    targetRevision: main
    path: kubernetes/manifests
```

Se il Capability Role dispone delle CodeCommit autorizzazioni richieste, il repository viene utilizzato direttamente e non è richiesta alcuna configurazione del repository. Per informazioni dettagliate, vedi the section called "Configura i repository".

Repository Git da CodeConnections:

```
spec:
  source:
    repoURL: https://codeconnections.region.amazonaws.com/git-http/account-
id/region/connection-id/owner/repository.git
    targetRevision: main
```

```
    path: kubernetes/manifests
```

Il formato dell'URL del repository è derivato dall'ARN della CodeConnections connessione. Se il Capability Role dispone delle CodeConnections autorizzazioni richieste ed è configurata una connessione, il repository viene utilizzato direttamente e non è richiesta alcuna configurazione del repository. Per informazioni dettagliate, vedi the section called "Configura i repository".

Personalizza:

```
spec:
  source:
    repoURL: https://github.com/example/kustomize-app
    targetRevision: main
    path: overlays/production
    kustomize:
      namePrefix: prod-
```

Politiche di sincronizzazione

Controlla il modo in cui Argo CD sincronizza le applicazioni.

Sincronizzazione manuale (impostazione predefinita):

Le applicazioni richiedono l'approvazione manuale per la sincronizzazione:

```
spec:
  syncPolicy: {}  # No automated sync
```

Attiva manualmente la sincronizzazione:

```
kubectl patch application guestbook -n argocd \
  --type merge \
  --patch '{"operation": {"initiatedBy": {"username": "admin"}, "sync": {}}}'
```

Sincronizzazione automatica:

Le applicazioni si sincronizzano automaticamente quando vengono rilevate modifiche a Git:

```
spec:
  syncPolicy:
    automated: {}
```

Self-healing:

Ripristina automaticamente le modifiche manuali al cluster:

```
spec:
  syncPolicy:
    automated:
      selfHeal: true
```

Se abilitato, Argo CD ripristina qualsiasi modifica manuale apportata direttamente al cluster, assicurando che Git rimanga la fonte della verità.

Potatura:

Elimina automaticamente le risorse rimosse da Git:

```
spec:
  syncPolicy:
    automated:
      prune: true
```

> **Warning**
>
> Il pruning eliminerà le risorse dal cluster. Usare con cautela negli ambienti di produzione.

Sincronizzazione automatica combinata:

```
spec:
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=true
```

Riprova la configurazione:

Configura il comportamento dei tentativi di ripetizione per le sincronizzazioni non riuscite:

```
spec:
```

```
  syncPolicy:
    retry:
      limit: 5  # Number of failed sync attempts; unlimited if less than 0
      backoff:
        duration: 5s  # Amount to back off (default unit: seconds, also supports "2m",
 "1h")
        factor: 2  # Factor to multiply the base duration after each failed retry
        maxDuration: 3m  # Maximum amount of time allowed for the backoff strategy
```

Ciò è particolarmente utile per le risorse che dipendono dalla creazione iniziale dei CRD o quando si lavora con istanze kro in cui il CRD potrebbe non essere immediatamente disponibile.

Opzioni di sincronizzazione

Configurazione di sincronizzazione aggiuntiva:

Crea lo spazio dei nomi se non esiste:

```
spec:
  syncPolicy:
    syncOptions:
    - CreateNamespace=true
```

Salta la corsa a secco per le risorse mancanti:

Utile quando si applicano risorse che dipendono da CRD che non esistono ancora (come le istanze kro):

```
spec:
  syncPolicy:
    syncOptions:
    - SkipDryRunOnMissingResource=true
```

Questo può essere applicato anche a risorse specifiche utilizzando un'etichetta sulla risorsa stessa.

Convalida le risorse prima di applicarle:

```
spec:
  syncPolicy:
    syncOptions:
    - Validate=true
```

Applica solo non sincronizzato:

```
spec:
  syncPolicy:
    syncOptions:
    - ApplyOutOfSyncOnly=true
```

Funzionalità di sincronizzazione avanzate

Argo CD supporta funzionalità di sincronizzazione avanzate per implementazioni complesse:

- Sync waves: controlla l'ordine di creazione delle risorse con le annotazioni `argocd.argoproj.io/sync-wave`
- Sync hooks: esegui i lavori prima o dopo la sincronizzazione con `argocd.argoproj.io/hook` le annotazioni (PreSync,,) PostSync SyncFail
- Valutazione dello stato delle risorse: controlli di integrità personalizzati per le risorse specifiche dell'applicazione

Per i dettagli, consultate Sync Waves e Resource Hooks nella documentazione del CD Argo.

Ignora le differenze

Impedisci ad Argo CD di sincronizzare campi specifici gestiti da altri controller (come HPA che gestisce le repliche):

```
spec:
  ignoreDifferences:
  - group: apps
    kind: Deployment
    jsonPointers:
    - /spec/replicas
```

Per i dettagli sugli schemi di ignoramento e sulle esclusioni dei campi, consultate Diffing Customization nella documentazione del CD Argo.

Multi-environment distribuzione

Implementa la stessa applicazione in più ambienti:

Sviluppo:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-dev
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: develop
    path: overlays/development
  destination:
    name: dev-cluster
    namespace: my-app
```

Produzione:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-prod
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: main
    path: overlays/production
  destination:
    name: prod-cluster
    namespace: my-app
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Monitora e gestisci le applicazioni

Visualizza lo stato dell'applicazione:

```
kubectl get application my-app -n argocd
```

Accedi all'interfaccia utente di Argo CD:

Apri l'interfaccia utente di Argo CD tramite la console EKS per visualizzare la topologia dell'applicazione, lo stato di sincronizzazione, lo stato delle risorse e la cronologia di distribuzione. Vedi the section called "Lavorare con Argo CD" per le istruzioni di accesso all'interfaccia utente.

Applicazioni di rollback:

Ripristina una revisione precedente utilizzando l'interfaccia utente di Argo CD, l'Argo CD CLI o aggiornando le specifiche `targetRevision` dell'applicazione a un commit o tag Git precedente.

Utilizzo dell'Argo CD CLI:

```
argocd app rollback argocd/my-app <revision-id>
```

> **Note**
>
> Quando si utilizza l'Argo CD CLI con la funzionalità gestita, specificare le applicazioni con il prefisso dello spazio dei nomi:. `namespace/appname`

Per ulteriori informazioni, consulta il rollback dell'app argocd nella documentazione di Argo CD.

Risorse aggiuntive

- the section called "Progetti"- Organizza le applicazioni con Projects per ambienti multi-tenant
- the section called "ApplicationSets"- Implementa su più cluster con modelli
- Specifiche dell'applicazione: riferimento completo all'API dell'applicazione
- Opzioni di sincronizzazione: configurazione di sincronizzazione avanzata

## Usa ApplicationSets

ApplicationSets genera più applicazioni da modelli, consentendoti di distribuire la stessa applicazione su più cluster, ambienti o namespace con un'unica definizione di risorsa.

Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD

- Accesso al repository configurato (vedi) the section called "Configura i repository"
- `kubectl`configurato per comunicare con il cluster

> **Note**
>
> Non sono necessari cluster di destinazione multipli per ApplicationSets. È possibile utilizzare generatori diversi dal generatore di cluster (come i generatori list, git o matrix) per distribuire applicazioni senza cluster remoti.

Come funziona ApplicationSets

ApplicationSets utilizzate i generatori per produrre parametri, quindi applicate tali parametri a un modello di applicazione. Ogni set di parametri generati crea un'applicazione.

Generatori comuni per le implementazioni EKS:

- Generatore di elenchi: definisce in modo esplicito cluster e parametri per ogni ambiente
- Generatore di cluster: implementa automaticamente su tutti i cluster registrati
- Generatore Git: genera applicazioni dalla struttura del repository
- Generatore di matrici: combina generatori per implementazioni multidimensionali
- Generatore di fusione: unisce i parametri di più generatori

Per un riferimento completo al generatore, consultate la Documentazione. ApplicationSet

Generatore di elenchi

Esegui la distribuzione su più cluster con configurazione esplicita:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook-all-clusters
  namespace: argocd
spec:
  generators:
  - list:
```

```
      elements:
      - environment: dev
        replicas: "2"
      - environment: staging
        replicas: "3"
      - environment: prod
        replicas: "5"
  template:
    metadata:
      name: 'guestbook-{{environment}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/guestbook
        targetRevision: HEAD
        path: 'overlays/{{environment}}'
      destination:
        name: '{{environment}}-cluster'
        namespace: guestbook
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

> **Note**
>
> Utilizzalo `destination.name` con i nomi dei cluster per una migliore leggibilità. ARNs Se necessario, il `destination.server` campo funziona anche con il cluster EKS.

Questo crea tre applicazioni: `guestbook-dev`guestbook-staging, eguestbook-prod.

Generatore di cluster

Implementa automaticamente su tutti i cluster registrati:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-addons
  namespace: argocd
spec:
```

```
  generators:
  - clusters: {}
  template:
    metadata:
      name: '{{name}}-addons'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/cluster-addons
        targetRevision: HEAD
        path: addons
      destination:
        server: '{{server}}'
        namespace: kube-system
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

Questo crea automaticamente un'applicazione per ogni cluster registrato.

Cluster di filtri:

Utilizza `matchLabels` per includere cluster specifici o `matchExpressions` per escludere cluster:

```
spec:
  generators:
  - clusters:
      selector:
        matchLabels:
          environment: production
        matchExpressions:
        - key: skip-appset
          operator: DoesNotExist
```

Generatori Git

I generatori Git creano applicazioni basate sulla struttura del repository:

- Generatore di directory: distribuisci ogni directory come applicazione separata (utile per i microservizi)
- Generatore di file: genera applicazioni da file di parametri (utile per le implementazioni multi-tenant)

Esempio: implementazione di microservizi

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: microservices
  namespace: argocd
spec:
  generators:
  - git:
      repoURL: https://github.com/example/microservices
      revision: HEAD
      directories:
      - path: services/*
  template:
    metadata:
      name: '{{path.basename}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/microservices
        targetRevision: HEAD
        path: '{{path}}'
      destination:
        name: my-cluster
        namespace: '{{path.basename}}'
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
        syncOptions:
        - CreateNamespace=true
```

Per i dettagli sui generatori Git e sulla configurazione basata su file, consultate Git Generator nella documentazione del CD Argo.

Generatore di matrici

Combina più generatori da implementare su più dimensioni (ambienti × cluster):

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
```

```
  name: multi-env-multi-cluster
  namespace: argocd
spec:
  generators:
  - matrix:
      generators:
      - list:
          elements:
          - environment: dev
          - environment: staging
          - environment: prod
      - clusters:
          selector:
            matchLabels:
              region: us-west-2
  template:
    metadata:
      name: 'app-{{environment}}-{{name}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/app
        targetRevision: HEAD
        path: 'overlays/{{environment}}'
      destination:
        name: '{{name}}'
        namespace: 'app-{{environment}}'
```

Per i dettagli sulla combinazione dei generatori, consultate Matrix Generator nella documentazione di Argo CD.

Implementazione in più regioni

Implementa su cluster in più regioni:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: global-app
  namespace: argocd
spec:
  generators:
  - list:
```

```
      elements:
      - clusterName: prod-us-west
        region: us-west-2
      - clusterName: prod-us-east
        region: us-east-1
      - clusterName: prod-eu-west
        region: eu-west-1
  template:
    metadata:
      name: 'app-{{region}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/app
        targetRevision: HEAD
        path: kubernetes
        helm:
          parameters:
          - name: region
            value: '{{region}}'
      destination:
        name: '{{clusterName}}'
        namespace: app
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

Gestisci ApplicationSets

Visualizza ApplicationSets e genera applicazioni:

```
kubectl get applicationsets -n argocd
kubectl get applications -n argocd -l argocd.argoproj.io/application-set-
name=<applicationset-name>
```

Aggiorna un ApplicationSet:

Modifica le ApplicationSet specifiche e riapplica. Argo CD aggiorna automaticamente tutte le applicazioni generate:

```
kubectl apply -f applicationset.yaml
```

Eliminare un ApplicationSet:

```
kubectl delete applicationset <name> -n argocd
```

> **Warning**
>
> Eliminazione ed ApplicationSet eliminazione di tutte le applicazioni generate. Se tali applicazioni lo hanno fatto`prune: true`, le relative risorse verranno eliminate anche dai cluster di destinazione.
> Per preservare le risorse distribuite durante l'eliminazione di un file ApplicationSet, impostate su `true` nelle `.syncPolicy.preserveResourcesOnDeletion` specifiche. ApplicationSet Per ulteriori informazioni, consultate Application Pruning & Resource Deletion nella documentazione del CD Argo.

> **Important**
>
> La ApplicationSets funzionalità di Argo CD prevede considerazioni sulla sicurezza di cui è necessario tenere conto prima di utilizzarla. ApplicationSets Per ulteriori informazioni, consultate la sezione ApplicationSet Sicurezza nella documentazione del CD Argo.

Risorse aggiuntive

- the section called “Progetti”- Organizza ApplicationSets con progetti
- the section called “Crea applicazioni”- Comprendi la configurazione dell'applicazione
- ApplicationSet Documentazione: riferimenti e modelli completi del generatore
- Generator Reference - Specifiche dettagliate del generatore

## Configurare l'accesso privato agli endpoint per Argo CD

Per impostazione predefinita, puoi accedere all'endpoint Argo CD tramite Internet pubblico. Puoi configurare l'accesso privato agli endpoint in modo che il traffico tra il tuo VPC e l'endpoint Argo CD rimanga all'interno AWS della rete, senza passare attraverso la rete Internet pubblica.

Quando abiliti l'accesso privato, AWS aggiorna il Domain Name System (DNS) in modo che il nome host dell'endpoint Argo CD si risolva in indirizzi IP privati associati al tuo endpoint VPC. È necessario

disporre della connettività di rete a questi IP privati (tramite il VPC, una connessione di rete privata virtuale (VPN) o AWS Direct Connect) per accedere all'endpoint Argo CD.

## Come funziona l'accesso privato

Quando configurate l'accesso privato per la funzionalità Argo CD:

1. Crei un endpoint VPC per il `com.amazonaws.`*`region-code`*`.eks-capabilities` servizio nel tuo VPC.
2. Fornisci l'ID dell'endpoint VPC durante la creazione o l'aggiornamento della funzionalità Argo CD.
3. AWS configura i record DNS in modo che il nome host dell'endpoint Argo CD si risolva in indirizzi IP privati associati all'endpoint VPC.
4. Tutto il traffico verso l'endpoint Argo CD fluisce privatamente attraverso l'endpoint VPC.

## Prerequisiti

- Un cluster Amazon Elastic Kubernetes Service (Amazon EKS) con funzionalità Argo CD attiva (vedi) the section called "Crea una funzionalità Argo CD"
- Familiarità con i concetti di Amazon VPC, tra cui sottoreti, gruppi di sicurezza ed endpoint VPC di interfaccia
- Un VPC con sottoreti nelle zone di disponibilità a cui desideri l'accesso privato
- AWS Versione AWS `2.12.3` CLI (Command Line Interface) o successiva

## Fase 1: Creare un endpoint VPC per eks-capabilities

Crea un endpoint VPC di interfaccia per il servizio. `com.amazonaws.`*`region-code`*`.eks-capabilities`

Console

1. Apri la Console Amazon VPC.
2. Nel riquadro di navigazione a sinistra, scegli Endpoints (Endpoint).
3. Seleziona Crea endpoint.
4. Per la categoria di servizio, scegli AWS servizi.
5. Nel campo di ricerca, inserisci `eks-capabilities` e selezionacom.`amazonaws.`*`region-code`*`.eks-capabilities`.

6. Per VPC, seleziona il VPC da cui desideri accedere all'endpoint Argo CD.
7. Per le sottoreti, selezionate una sottorete in ogni zona di disponibilità a cui desiderate l'accesso privato.
8. Per i gruppi di sicurezza, seleziona un gruppo di sicurezza che consenta il traffico HTTPS (porta 443) in entrata dalle reti client.
9. Seleziona Crea endpoint.
10. Prendi nota dell'ID dell'endpoint VPC (ad esempio,). `vpce-0123456789abcdef0`

CLI

Sostituiscilo *region-code* con la tua AWS regione, *vpc-id* con il tuo ID VPC, *subnet-id-1* *subnet-id-2* con ID di sottorete in diverse zone di disponibilità e *sg-id* con un gruppo di sicurezza che consente il traffico HTTPS in entrata.

```
aws ec2 create-vpc-endpoint \
  --region region-code \
  --vpc-endpoint-type Interface \
  --service-name com.amazonaws.region-code.eks-capabilities \
  --vpc-id vpc-id \
  --subnet-ids subnet-id-1 subnet-id-2 \
  --security-group-ids sg-id
```

Prendere nota dell'ID `VpcEndpointId` nella risposta.

> **Important**
>
> Assicurati che il gruppo di sicurezza collegato all'endpoint VPC consenta il traffico HTTPS in entrata (porta 443). La fonte di traffico deve includere tutte le reti in cui gli utenti o le applicazioni accedono all'endpoint Argo CD.

## Fase 2: Configurare l'accesso privato sulla funzionalità Argo CD

Una volta che l'endpoint VPC è disponibile, configura la funzionalità Argo CD per utilizzarlo. È possibile configurare l'accesso privato quando si crea una nuova funzionalità o si aggiorna una funzionalità esistente.

Crea una nuova funzionalità con accesso privato

Includi il `networkAccess` nel `create-capability` comando. Sostituisci *region-code* *my-cluster*, e altri segnaposto con i tuoi valori:

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --type ARGOCD \
  --role-arn arn:aws:iam::111122223333:role/ArgoCDCapabilityRole \
  --configuration '{
    "argoCd": {
      "awsIdc": {
        "idcInstanceArn": "arn:aws:sso:::instance/ssoins-1234567890abcdef",
        "idcRegion": "idc-region-code"
      },
      "rbacRoleMappings": [{
        "role": "ADMIN",
        "identities": [{
          "id": "user-id",
          "type": "SSO_USER"
        }]
      }],
      "networkAccess": {
        "vpceIds": ["vpce-0123456789abcdef0"]
      }
    }
  }'
```

Aggiorna una funzionalità esistente per abilitare l'accesso privato

Sostituisci *region-code* *my-cluster*, *my-argocd*, e l'ID dell'endpoint VPC con i tuoi valori:

```
aws eks update-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --configuration '{
    "argoCd": {
      "networkAccess": {
        "vpceIds": ["vpce-0123456789abcdef0"]
      }
```

```
    }
  }'
```

Attendi il completamento dell'aggiornamento della funzionalità:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.status' \
  --output text
```

Lo stato ritorna al `ACTIVE` termine dell'aggiornamento.

## Fase 3: Verifica della risoluzione e della connettività DNS

Dopo che la funzionalità diventa attiva con l'accesso privato configurato, verifica che il nome host dell'endpoint Argo CD si risolva in indirizzi IP privati.

Esegui il seguente `nslookup` comando da qualsiasi rete:

```
nslookup your-argocd-endpoint.eks-capabilities.region-code.amazonaws.com
```

La risposta dovrebbe mostrare indirizzi IP privati (ad esempio, indirizzi negli intervalli RFC 1918 come `10.x.x.x` o`172.16.x.x`).

Se l'endpoint non si risolve in IP privati, verifica che:

- Lo stato della funzionalità è `ACTIVE` (AWS crea record DNS dopo che la funzionalità è diventata attiva)
- L'endpoint VPC è nello stato `available`
- L'ID dell'endpoint VPC nella configurazione delle funzionalità corrisponde all'endpoint che hai creato

> **Note**
>
> Dopo aver configurato l'accesso privato, il nome host dell'endpoint Argo CD si risolve in indirizzi IP privati. Per connetterti, devi disporre di connettività di rete a quegli IP privati tramite il tuo VPC, una connessione VPN o AWS Direct Connect.

### Disabilita l'accesso privato

Per disabilitare l'accesso privato e ripristinare l'accesso pubblico agli endpoint, aggiorna la capacità di rimuovere gli ID degli endpoint VPC:

```
aws eks update-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --configuration '{
    "argoCd": {
      "networkAccess": {
        "vpceIds": []
      }
    }
  }'
```

### Considerazioni

- Dopo aver abilitato l'accesso privato, l'endpoint Argo CD è raggiungibile solo dall'interno del VPC o dalle reti ad esso collegate (ad esempio, tramite VPN o Direct Connect). AWS
- L'endpoint VPC deve rimanere nello `available` stato per consentire l'accesso privato al funzionamento. Se si elimina l'endpoint VPC, si perde l'accesso all'endpoint Argo CD finché non si crea un nuovo endpoint e si aggiorna la funzionalità.
- AWS attualmente non supporta le policy degli endpoint VPC per l'endpoint del `eks-capabilities` servizio.

## Considerazioni su Argo CD

Questo argomento tratta importanti considerazioni sull'utilizzo del CD EKS Capability for Argo, tra cui pianificazione, autorizzazioni, autenticazione e modelli di distribuzione multi-cluster.

## Pianificazione

Prima di distribuire Argo CD, considera quanto segue:

Strategia di deposito: determina dove verranno archiviati i manifesti dell'applicazione (CodeCommit,, GitHub, GitLab Bitbucket). Pianifica la struttura del tuo repository e la strategia di ramificazione per diversi ambienti.

Strategia RBAC: pianifica quali team o utenti devono avere accesso come amministratore, editor o visualizzatore. Associali ai gruppi AWS Identity Center o ai ruoli Argo CD.

Multi-cluster architettura: Determina se gestirai più cluster da una singola istanza Argo CD. Prendi in considerazione l'utilizzo di un cluster di gestione dedicato per Argo CD.

Organizzazione delle applicazioni: pianifica come strutturerai le applicazioni e ApplicationSets. Prendi in considerazione l'utilizzo di progetti per organizzare le applicazioni per team o ambiente.

Criteri di sincronizzazione: decidi se le applicazioni devono essere sincronizzate automaticamente o richiedere l'approvazione manuale. La sincronizzazione automatica è comune per lo sviluppo, manuale per la produzione.

## Permissions

Per informazioni dettagliate sui ruoli di capacità di IAM, sulle politiche di fiducia e sulle best practice di sicurezza, consulta the section called “Capacità, ruolo IAM” ethe section called “Considerazioni sulle funzionalità EKS”.

Panoramica del ruolo di capacità IAM

Quando crei una risorsa di funzionalità Argo CD, fornisci un ruolo di capacità IAM. A differenza di ACK, Argo CD gestisce principalmente le risorse Kubernetes, non le risorse direttamente. AWS Tuttavia, l'IAM Capability Role è richiesto per:

- Accesso ai repository Git privati in CodeCommit
- Integrazione con AWS Identity Center per l'autenticazione
- Accesso ai AWS segreti in Secrets Manager (se configurato)
- Cross-cluster implementazioni su altri cluster EKS

CodeCommit integrazione

Se utilizzi i CodeCommit repository, allega una policy con autorizzazioni di lettura:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ]
}
```

**Important**

Per l'uso in produzione, limita il `Resource` campo a specifici ARN del repository invece di utilizzarlo. `"*"`
Esempio:

```
"Resource": "arn:aws:codecommit:us-west-2:111122223333:my-app-repo"
```

Ciò limita l'accesso della funzionalità Argo CD ai soli repository che deve gestire.

### Integrazione di Secrets Manager

Se stai archiviando le credenziali del repository in Secrets Manager, allega la policy gestita per l'accesso in lettura:

```
arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess
```

Questa politica include le autorizzazioni necessarie:`secretsmanager:GetSecretValue`, e le autorizzazioni di `secretsmanager:DescribeSecret` decrittografia KMS.

### Configurazione di base

Per le funzionalità di base di Argo CD con repository Git pubblici, non sono richieste politiche IAM aggiuntive oltre alla policy di fiducia.

## Autenticazione

### AWS Integrazione con Identity Center

La funzionalità gestita da Argo CD si integra direttamente con AWS Identity Center (precedentemente AWS SSO), consentendoti di utilizzare il tuo provider di identità esistente per l'autenticazione.

Quando si configura l'integrazione con Identity Center: AWS

1. Gli utenti accedono all'interfaccia utente di Argo CD tramite la console EKS
2. Si autenticano utilizzando AWS Identity Center (che può essere federato con il provider di identità aziendale)
3. AWS Identity Center fornisce informazioni su utenti e gruppi ad Argo CD
4. Argo CD associa utenti e gruppi ai ruoli RBAC in base alla configurazione
5. Gli utenti vedono solo le applicazioni e le risorse a cui hanno il permesso di accedere

### Semplificazione dell'accesso con i set di autorizzazioni di Identity Center

AWS Identity Center fornisce due percorsi di autenticazione distinti quando si lavora con Argo CD:

Autenticazione API Argo CD: Identity Center fornisce l'autenticazione SSO all'interfaccia utente e all'API Argo CD. Questo è configurato tramite le mappature dei ruoli RBAC della funzionalità Argo CD.

Accesso al cluster EKS: la funzionalità Argo CD utilizza il ruolo IAM fornito dal cliente per autenticarsi con i cluster EKS tramite voci di accesso. Queste voci di accesso possono essere configurate manualmente per aggiungere o rimuovere autorizzazioni.

È possibile utilizzare i set di autorizzazioni di Identity Center per semplificare la gestione delle identità consentendo a una singola identità di accedere ai cluster Argo CD ed EKS. Ciò riduce il sovraccarico richiedendo la gestione di una sola identità su entrambi i sistemi, anziché mantenere credenziali separate per l'accesso al CD Argo e l'accesso al cluster.

### Mappature dei ruoli RBAC

Argo CD ha ruoli integrati che puoi mappare agli utenti e ai gruppi di AWS Identity Center:

ADMIN: accesso completo a tutte le applicazioni e le impostazioni. Può creare, aggiornare ed eliminare applicazioni. Può gestire la configurazione del CD Argo.

EDITOR: Può creare e modificare applicazioni. Non è possibile modificare le impostazioni del CD Argo o eliminare applicazioni.

VIEWER: Read-only accesso alle applicazioni. Può visualizzare lo stato e la cronologia delle applicazioni. Non è possibile apportare modifiche.

> **Note**
>
> I nomi dei ruoli fanno distinzione tra maiuscole e minuscole e devono essere in maiuscolo (ADMIN, EDITOR, VIEWER).

> **Important**
>
> L'integrazione di EKS Capabilities con AWS Identity Center supporta fino a 1.000 identità per funzionalità Argo CD. Un'identità può essere un utente o un gruppo.

## Multi-cluster distribuzioni

La funzionalità gestita di Argo CD supporta le implementazioni multi-cluster, consentendo di gestire le applicazioni tra cluster di sviluppo, staging e produzione da una singola istanza Argo CD.

Come funziona il multi-cluster

Quando si registrano cluster aggiuntivi con Argo CD:

1. Crei i segreti dei cluster che fanno riferimento ai cluster EKS di destinazione tramite ARN
2. Si creano applicazioni o destinate a ApplicationSets cluster diversi
3. Argo CD si connette a ciascun cluster per distribuire e controllare le risorse
4. Puoi visualizzare e gestire tutti i cluster da un'unica interfaccia utente di Argo CD

Prerequisiti per il multi-cluster

Prima di registrare cluster aggiuntivi:

- Crea una voce di accesso sul cluster di destinazione per il ruolo di funzionalità Argo CD
- Garantisci la connettività di rete tra la funzionalità Argo CD e i cluster di destinazione

- Verifica le autorizzazioni IAM per accedere ai cluster di destinazione

### Registra un cluster

Registra i cluster utilizzando Kubernetes Secrets nel namespace. `argocd`

Ottieni l'ARN del cluster di destinazione. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster di destinazione e sostituiscilo *`target-cluster`* con il nome del cluster di destinazione.

```
aws eks describe-cluster \
  --region region-code \
  --name target-cluster \
  --query 'cluster.arn' \
  --output text
```

Crea un cluster segreto utilizzando l'ARN del cluster:

```
apiVersion: v1
kind: Secret
metadata:
  name: target-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
type: Opaque
stringData:
  name: target-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/target-cluster
  project: default
```

> **Important**
>
> Utilizza l'ARN del cluster EKS nel `server` campo, non l'URL del server API Kubernetes. La funzionalità gestita richiede che gli ARN identifichino i cluster di destinazione.

Applica il segreto:

```
kubectl apply -f cluster-secret.yaml
```

### Configurare Access Entry sul cluster di destinazione

Il cluster di destinazione deve disporre di una voce di accesso che conceda al ruolo di capacità Argo CD l'autorizzazione per distribuire le applicazioni. Sostituisci *region-code* con la AWS regione in cui si trova il cluster di destinazione, sostituisci *target-cluster* con il nome del cluster di destinazione e sostituisci l'ARN con il tuo ruolo di funzionalità Argo CD ARN.

```
aws eks create-access-entry \
  --region region-code \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::111122223333:role/ArgoCDCapabilityRole \
  --type STANDARD \
  --kubernetes-groups system:masters
```

> **Note**
>
> Per l'uso in produzione, prendi in considerazione l'utilizzo di gruppi Kubernetes più restrittivi anziché. `system:masters`

### Accesso privato al cluster

La funzionalità gestita da Argo CD può essere implementata su cluster EKS completamente privati senza richiedere peering VPC o configurazioni di rete specializzate. AWS gestisce automaticamente la connettività tra la funzionalità Argo CD e i cluster remoti privati. Assicurati che i controlli di accesso al repository e le policy RBAC di Argo CD siano configurati correttamente.

### Cross-account implementazioni

Per le distribuzioni su più account, aggiungi il ruolo Argo CD IAM Capability dall'account di origine all'EKS Access Entry del cluster di destinazione:

1. Nell'account di destinazione, crea una voce di accesso sul cluster EKS di destinazione
2. Utilizzate l'Argo CD IAM Capability Role ARN dell'account di origine come principale
3. Configura le autorizzazioni RBAC Kubernetes appropriate per l'Access Entry
4. Registra il cluster di destinazione in Argo CD utilizzando il relativo cluster EKS ARN

Non è richiesta alcuna creazione aggiuntiva di ruoli IAM o configurazione di policy di trust: EKS Access Entries gestisce l'accesso tra account.

## Best practice

Usa fonti dichiarative come fonte di verità: archivia tutti i manifesti delle tue applicazioni in fonti dichiarative (repository Git, registri Helm o immagini OCI), abilitando il controllo della versione, gli audit trail e la collaborazione.

Implementa un RBAC adeguato: utilizza l'integrazione con AWS Identity Center per controllare chi può accedere e gestire le applicazioni in Argo CD. Argo CD supporta il controllo granulare degli accessi alle risorse all'interno delle applicazioni (Deployments, Pods, Secrets). ConfigMaps

Utilizzo ApplicationSets per implementazioni multiambiente: da utilizzare per distribuire applicazioni su più cluster o ApplicationSets namespace con configurazioni diverse.

## Gestione del ciclo di vita

Politiche di sincronizzazione delle applicazioni

Controlla il modo in cui Argo CD sincronizza le applicazioni:

Sincronizzazione manuale: le applicazioni richiedono l'approvazione manuale per sincronizzare le modifiche. Consigliato per ambienti di produzione.

Sincronizzazione automatica: le applicazioni si sincronizzano automaticamente quando vengono rilevate modifiche a Git. Comune negli ambienti di sviluppo e staging.

Self-healing: ripristina automaticamente le modifiche manuali apportate al cluster. Assicura che lo stato del cluster corrisponda a Git.

Eliminazione: elimina automaticamente le risorse rimosse da Git. Usalo con cautela in quanto può eliminare risorse.

Integrità delle applicazioni

Argo CD monitora continuamente lo stato delle applicazioni:

- Sano: tutte le risorse funzionano come previsto
- Progresso: le risorse vengono create o aggiornate
- Degradato: alcune risorse non sono integre
- Sospesa: l'applicazione è in pausa
- Mancante: mancano delle risorse dal cluster

### Sincronizza finestre

Configura le finestre di sincronizzazione per controllare quando le applicazioni possono essere sincronizzate:

- Consenti le sincronizzazioni solo durante le finestre di manutenzione
- Blocca le sincronizzazioni durante l'orario lavorativo
- Pianifica le sincronizzazioni automatiche per orari specifici
- Usa le finestre di sincronizzazione nelle situazioni in cui devi apportare modifiche e interrompere qualsiasi sincronizzazione (scenari rivoluzionari)

## Configurazione Webhook per una sincronizzazione più rapida

Per impostazione predefinita, Argo CD esegue il polling dei repository Git ogni 6 minuti per rilevare le modifiche. Per implementazioni più reattive, configura i webhook Git in modo che attivino sincronizzazioni immediate quando vengono inviate modifiche.

I webhook offrono diversi vantaggi:

- Risposta di sincronizzazione immediata quando viene inviato il codice (secondi anziché minuti)
- Riduzione del sovraccarico di polling e miglioramento delle prestazioni del sistema
- Uso più efficiente dei limiti di velocità delle API
- Migliore esperienza utente con un feedback più rapido

### Endpoint Webhook

L'URL del webhook segue lo schema`${serverUrl}/api/webhook`, `serverUrl` dov'è l'URL del server CD Argo. Per trovare l'URL del tuo server, consulta. the section called “URL dell'endpoint Argo CD”

Ad esempio, se l'URL del server CD Argo è`https://my-argocd-dc855fdf-111122223333.eks-capabilities.us-west-2.amazonaws.com`, l'URL del webhook è:

```
https://my-argocd-dc855fdf-111122223333.eks-capabilities.us-west-2.amazonaws.com/api/webhook
```

Configura i webhook in base al provider Git

GitHub: Nelle impostazioni del tuo repository, aggiungi un webhook con l'URL del webhook Argo CD. Imposta il tipo di contenuto su `application/json` e seleziona «Solo l'evento push».

GitLab: Nelle impostazioni del progetto, aggiungete un webhook con l'URL del webhook Argo CD. Abilita «Eventi push» e opzionalmente «Tag push events».

Bitbucket: nelle impostazioni del tuo repository, aggiungi un webhook con l'URL del webhook Argo CD. Seleziona «Repository push» come trigger.

CodeCommit: crea una EventBridge regola Amazon che attiva le modifiche allo stato del CodeCommit repository e invia notifiche all'endpoint webhook Argo CD.

Per istruzioni dettagliate sulla configurazione del webhook, consulta Argo CD Webhook Configuration. https://argo-cd.readthedocs.io/en/stable/operator-manual/webhook/

> **Note**
>
> I webhook completano il polling, non lo sostituiscono. Argo CD continua a eseguire il polling dei repository come meccanismo di riserva nel caso in cui le notifiche dei webhook non vengano rilevate.

### Fasi successive

- the section called "Lavorare con Argo CD"- Scopri come creare e gestire le applicazioni Argo CD
- the section called "Risoluzione dei problemi"- Risolvi i problemi relativi a Argo CD
- the section called "Lavorare con le funzionalità"- Gestisci le tue risorse di funzionalità Argo CD

## Risolvi i problemi relativi alle funzionalità Argo CD

> **Note**
>
> Le funzionalità EKS sono completamente gestite ed eseguite all'esterno del cluster. Non hai accesso diretto ai namespace dei controller. È possibile configurare la distribuzione dei log del controller per la visibilità del comportamento del controller. Consulta the section called

"Funzionalità, registri dei controller". La risoluzione dei problemi si concentra sullo stato delle funzionalità, sullo stato dell'applicazione e sulla configurazione.

## La funzionalità è ATTIVA ma le applicazioni non si sincronizzano

Se la funzionalità del CD Argo mostra `ACTIVE` lo stato ma le applicazioni non si sincronizzano, controllate lo stato della funzionalità e l'applicazione.

Verifica lo stato della funzionalità:

È possibile visualizzare i problemi relativi all'integrità e allo stato delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la  scheda  Funzionalità per visualizzare lo stato e lo stato di tutte le funzionalità.

AWS CLI:

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd

# Look for issues in the health section
```

Cause comuni:

- Repository non configurato: repository Git non aggiunto al CD Argo
- Autenticazione non riuscita: chiave SSH, token o credenziali non validi CodeCommit
- Applicazione non creata: nel cluster non sono presenti risorse applicative

- Politica di sincronizzazione: è richiesta la sincronizzazione manuale (la sincronizzazione automatica non è abilitata)
- Autorizzazioni IAM: autorizzazioni mancanti per CodeCommit o Secrets Manager

Controlla lo stato della domanda:

```
# List applications
kubectl get application -n argocd

# View sync status
kubectl get application my-app -n argocd -o jsonpath='{.status.sync.status}'

# View application health
kubectl get application my-app -n argocd -o jsonpath='{.status.health}'
```

Verifica le condizioni della domanda:

```
# Describe application to see detailed status
kubectl describe application my-app -n argocd

# View application health
kubectl get application my-app -n argocd -o jsonpath='{.status.health}'
```

## Applicazioni bloccate nello stato «In corso»

Se un'applicazione viene visualizzata `Progressing` ma non viene mai visualizzata`Healthy`, controlla lo stato delle risorse e gli eventi dell'applicazione.

Controlla lo stato delle risorse:

```
# View application resources
kubectl get application my-app -n argocd -o jsonpath='{.status.resources}'

# Check for unhealthy resources
kubectl describe application my-app -n argocd | grep -A 10 "Health Status"
```

Cause comuni:

- Implementazione non pronta: i pod non si avviano o le sonde di prontezza non funzionano
- Dipendenze tra le risorse: risorse in attesa che altre risorse siano pronte

- Errori di estrazione delle immagini: immagini del contenitore non accessibili
- Risorse insufficienti: il cluster non dispone di CPU o memoria per i pod

Verifica la configurazione del cluster di destinazione (per configurazioni multi-cluster):

```
# List registered clusters
kubectl get secret -n argocd -l argocd.argoproj.io/secret-type=cluster

# View cluster secret details
kubectl get secret cluster-secret-name -n argocd -o yaml
```

## Errori di autenticazione del repository

Se Argo CD non può accedere ai tuoi repository Git, verifica la configurazione di autenticazione.

Per CodeCommit i repository:

Verifica che il ruolo di capacità IAM disponga delle CodeCommit autorizzazioni:

```
# View IAM policies
aws iam list-attached-role-policies --role-name my-argocd-capability-role
aws iam list-role-policies --role-name my-argocd-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-argocd-capability-role --policy-name policy-name
```

Il ruolo richiede l'`codecommit:GitPull`autorizzazione per i repository.

Per i repository Git privati:

Verifica che le credenziali del repository siano configurate correttamente:

```
# Check repository secret exists
kubectl get secret -n argocd repo-secret-name -o yaml
```

Assicurati che il segreto contenga le credenziali di autenticazione corrette (chiave SSH, token o). username/password

Per i repository che utilizzano Secrets Manager:

```
# Verify IAM Capability Role has Secrets Manager permissions
aws iam list-attached-role-policies --role-name my-argocd-capability-role

# Test secret retrieval
aws secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:region-
code:111122223333:secret:my-secret
```

## Multi-cluster problemi di distribuzione

Se le applicazioni non vengono distribuite su cluster remoti, verifica la registrazione del cluster e la configurazione dell'accesso.

Controlla la registrazione del cluster:

```
# List registered clusters
kubectl get secret -n argocd -l argocd.argoproj.io/secret-type=cluster

# Verify cluster secret format
kubectl get secret CLUSTER_SECRET_NAME -n argocd -o yaml
```

Assicurati che il `server` campo contenga l'ARN del cluster EKS, non l'URL dell'API Kubernetes.

Verifica la voce di accesso al cluster di destinazione:

Nel cluster di destinazione, verifica che il ruolo Argo CD Capability abbia una voce di accesso:

```
# List access entries (run on target cluster or use AWS CLI)
aws eks list-access-entries --cluster-name target-cluster

# Describe specific access entry
aws eks describe-access-entry \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::111122223333:role/my-argocd-capability-role
```

Controlla le autorizzazioni IAM per più account:

Per le distribuzioni su più account, verifica che il ruolo Argo CD Capability abbia una voce di accesso nel cluster di destinazione. La funzionalità gestita utilizza EKS Access Entries per l'accesso tra più account, non l'assunzione del ruolo IAM.

Per ulteriori informazioni sulla configurazione multi-cluster, vedere. the section called "Registra i cluster"

## Maggiore tempo di sincronizzazione delle applicazioni

Se la sincronizzazione delle applicazioni richiede più tempo del previsto, utilizza i seguenti passaggi diagnostici per identificare la causa.

### Controlla l'ora dell'ultima sincronizzazione

Conferma il ritardo controllando l'ultima volta che le applicazioni sono state sincronizzate:

```
# View last sync time for all applications
kubectl get application -n argocd -o jsonpath='{range .items[*]}{.metadata.name}{"\t"}
{.status.operationState.finishedAt}{"\n"}{end}'

# View last sync time for a specific application
kubectl get application my-app -n argocd -o
 jsonpath='{.status.operationState.finishedAt}'
```

### Verifica le condizioni dell'applicazione

Esamina le condizioni per la presentazione delle domande relative ai ritardi nelle code di riconciliazione:

```
# Check conditions on an application
kubectl get application my-app -n argocd -o jsonpath='{.status.conditions}'
```

### Controlla la configurazione di TargetRevision

Le applicazioni che utilizzano `targetRevision: HEAD` invalidano la cache del manifesto a ogni commit nel repository, il che rallenta i tempi di sincronizzazione:

```
# List applications using HEAD as targetRevision
kubectl get application -n argocd -o jsonpath='{range .items[?
(@.spec.source.targetRevision=="HEAD")]}{.metadata.name}{"\n"}{end}'
```

### Cause comuni

- Nessuna configurazione webhook: senza webhook, Argo CD esegue il polling dei repository con l'intervallo predefinito di 6 minuti. Ciò ritarda il rilevamento di nuovi commit.

- TargetRevision impostato su HEAD: ogni commit nel repository invalida la cache del manifesto. Argo CD rigenera quindi i manifest a ogni riconciliazione.
- Repository Git grandi o complessi: Monorepos o grafici Helm complessi causano una generazione lenta di manifest a causa del volume di file e modelli da elaborare.
- Numero elevato di risorse Kubernetes in una singola applicazione: le applicazioni che gestiscono molte risorse causano una lenta sincronizzazione della cache del cluster perché Argo CD deve tenere traccia dello stato di ciascuna risorsa.

Mitigazioni

- Configura i webhook Git: i webhook notificano immediatamente ad Argo CD quando vengono apportate modifiche, bypassando l'intervallo di polling predefinito. Per i passaggi di configurazione, consulta. the section called "Considerazioni su Argo CD"
- Usa nomi di ramo specifici o esegui il commit degli SHA: imposta `targetRevision` un nome di ramo o esegui il commit SHA invece di `HEAD` conservare la cache del manifesto tra le sincronizzazioni.
- Dividi monorepo di grandi dimensioni: dividi i repository di grandi dimensioni in repository più piccoli e mirati per ridurre i tempi di generazione dei manifest.
- Riduci le risorse per applicazione: suddividi le applicazioni con molte risorse Kubernetes in più applicazioni più piccole per ridurre i tempi di sincronizzazione della cache del cluster.
- Abilita la distribuzione dei log dei controller: i log dei controller forniscono visibilità sul comportamento di riconciliazione e sull'elaborazione delle code. Per le fasi di configurazione, vedere. the section called "Funzionalità, registri dei controller"

## Applicazioni che si sincronizzano ripetutamente o non sono sincronizzate

Se l'applicazione si sincronizza e poi viene eseguita immediatamente`OutOfSync`, o se rimane bloccata in un ciclo di sincronizzazione, la causa è solitamente una deriva tra ciò che Git definisce e ciò che esiste nel cluster. Inizia con la diagnostica di base.

Raccogli informazioni diagnostiche

```
# View current sync and health status
argocd app get my-app

# Show exact fields that differ between Git and live state
```

```
argocd app diff my-app

# Check whether the app has ever reached a stable state
argocd app history my-app
```

Il `argocd app diff` comando è il punto di partenza più utile. Mostra esattamente quali campi fanno apparire l'applicazione non sincronizzata.

Self-managed i certificati causano una deriva

Controller come cert-manager, OPA Gatekeeper e KEDA generano certificati in fase di esecuzione. Questi valori di runtime non sono in Git, quindi Argo CD rileva una deriva in ogni riconciliazione.

I sintomi sono:

- L'applicazione si sincronizza, quindi viene visualizzata immediatamente `OutOfSync`
- Il diff mostra le modifiche su un campo webhook o su un `caBundle` campo segreto TLS `data`

Per risolvere questo problema, aggiungi `ignoreDifferences` i campi interessati e abilita le opzioni `RespectIgnoreDifferences` di sincronizzazione:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
spec:
  ignoreDifferences:
    - group: admissionregistration.k8s.io
      kind: ValidatingWebhookConfiguration
      jsonPointers:
        - /webhooks/0/clientConfig/caBundle
    - group: ""
      kind: Secret
      jsonPointers:
        - /data/tls.crt
        - /data/tls.key
  syncPolicy:
    syncOptions:
      - RespectIgnoreDifferences=true
```

Self-heal interrompe i carichi di lavoro ad avvio lento

Quando `selfHeal` è abilitato, Argo CD risincronizza l'applicazione quando rileva una deriva. Se l'avvio del carico di lavoro impiega 30-60 secondi, l'autoriparazione si attiva prima che il carico di lavoro diventi effettivo. `Healthy` `prune`Se abilitato, questo potrebbe ridurre le risorse parzialmente avviate.

Per risolvere questo problema, correggi innanzitutto la deriva sottostante (vedi lo scenario del certificato). Se la deriva non è la causa, valuta la possibilità di disabilitare la correzione automatica per i carichi di lavoro che gestisci esclusivamente tramite Git:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
spec:
  syncPolicy:
    automated:
      selfHeal: false
      prune: false
```

> **Note**
>
> Self-heal il backoff timing è un'impostazione del controller a livello di istanza. Se hai bisogno di regolare la tempistica di autoriparazione anziché disabilitarla, apri una richiesta di assistenza. AWS

ApplicationSet o collisioni tra proprietà delle risorse

Se due applicazioni ApplicationSets gestiscono la stessa risorsa Kubernetes, Argo CD mostra un. `SharedResourceWarning` La risorsa non raggiunge mai uno stato stabile. Ciò si verifica in genere quando il nome di una risorsa condivisa non è definito per ambiente o cluster.

Per risolvere questo problema:

- Rendi unica la risorsa contenuta per proprietario. Aggiungi un suffisso di ambiente o cluster al nome della risorsa.
- Quando si rinomina un ApplicationSet, impostalo per `preserveResourcesOnDeletion: true` primo per evitare lo smontaggio distruttivo delle risorse esistenti:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: my-appset
spec:
  syncPolicy:
    preserveResourcesOnDeletion: true
```

Eliminazione bloccata dai finalizzatori di risorse

Se un'applicazione è bloccata o mostra «N oggetti rimanenti da eliminare», il `resources-finalizer.argocd.argoproj.io` finalizzatore blocca la rimozione fino all'eliminazione di tutte le risorse gestite. `Terminating` Una risorsa gestita con un proprio finalizzatore non elaborabile blocca l'eliminazione a tempo indeterminato.

Per confermare, elenca le risorse che hanno un timestamp di eliminazione ma non sono state rimosse:

```
kubectl get all -n my-namespace -o json | \
  jq '.items[] | select(.metadata.deletionTimestamp != null) | {name: .metadata.name,
 kind: .kind, finalizers: .metadata.finalizers}'
```

Per risolvere questo problema:

- Assicurati che il controller che possiede il finalizzatore di blocco sia integro e funzionante.
- Se il controller proprietario è integro ma il finalizzatore non è in fase di elaborazione, rimuovi il finalizzatore di blocco dalla risorsa bloccata.

Usando l'elenco dei finalizzatori stampato con il comando precedente, imposta l'elenco sui finalizzatori che desideri conservare e tralascia solo quello di blocco. Sostituisci `finalizer-a` e con questi nomi`finalizer-b`:

```
kubectl patch resource-kind
            resource-name -n my-namespace \
  --type merge -p '{"metadata":{"finalizers":["finalizer-a","finalizer-b"]}}'
```

Se il finalizzatore di blocco è l'unico sulla risorsa, passa un elenco vuoto:. `--type merge -p '{"metadata":{"finalizers":[]}}'`

> **Warning**
>
> Imposta l'elenco dei finalizzatori in modo esplicito anziché rimuovere una voce per posizione. Una risorsa può trasportare finalizzatori da più di un controller. I finalizzatori non sono in un ordine garantito. La rimozione della prima voce può eliminare il finalizzatore sbagliato, lasciando in posizione quello di blocco e la risorsa ancora bloccata.
> La rimozione di un finalizzatore consente inoltre di ignorare qualsiasi operazione di pulizia eseguita dal controller proprietario, con conseguente perdita di AWS risorse nel cluster senza alcuna registrazione. Esegui questa operazione solo dopo aver verificato che il controller proprietario non è in grado di elaborare il finalizzatore.

La sincronizzazione non riuscita non comporta il riavvio automatico della stessa revisione

Se una sincronizzazione con una revisione specifica fallisce, Argo CD non riprova automaticamente la stessa revisione. Ciò accade comunemente a causa di un difetto manifesto, ad esempio una chiave di variabile di ambiente `ComparisonError` duplicata.

Conferma controllando lo stato dell'applicazione:

```
argocd app get my-app
# Look for: Operation: Sync  Phase: Failed  Revision: <sha>
```

Per risolvere questo problema, correggi il difetto manifesto nel tuo repository Git e invia un nuovo commit. In alternativa, attiva una sincronizzazione manuale:

```
argocd app sync my-app
```

Il commit churn di Monorepo attiva un'ampia rigenerazione

Se molte applicazioni vengono registrate `HEAD` sullo stesso repository, qualsiasi commit in quel repository cambia per tutte le applicazioni. `HEAD` Ciò attiva la rigenerazione del manifesto per ogni applicazione, anche per quelle i cui file non sono stati modificati. Per ulteriori informazioni sulla memorizzazione nella cache `targetRevision` e sulla memorizzazione nella cache, vedere la sezione «Tempo di sincronizzazione delle applicazioni aumentato» in questa pagina.

Per limitare la rigenerazione solo ai file utilizzati da ciascuna applicazione, aggiungete l'`manifest-generate-paths`annotazione:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  annotations:
    argocd.argoproj.io/manifest-generate-paths: /apps/my-app
spec:
  source:
    repoURL: https://github.com/my-org/my-monorepo.git
    targetRevision: HEAD
    path: apps/my-app
```

Con questa annotazione, Argo CD rigenera i manifesti solo quando i file nel percorso specificato cambiano. Per le librerie condivise utilizzate tra le applicazioni, è possibile specificare più percorsi separati da punto e virgola (). `;`

Ove possibile, aggiungi `targetRevision` il nome o il tag di una filiale invece di. `HEAD`

I webhook predefiniti e mutanti di Kubernetes causano differenze fantasma

Se la tua applicazione viene visualizzata `OutOfSync` subito dopo una sincronizzazione, controlla le differenze per i campi che non hai mai impostato (come,, o). `terminationGracePeriodSeconds` `dnsPolicy` `/spec/replicas` Il server API Kubernetes o un webhook mutante hanno aggiunto questi campi al momento dell'applicazione.

Per risolvere questo problema per i campi gestiti da un altro controller (ad esempio `/spec/replicas` quando un HPA gestisce la scalabilità), aggiungi: `ignoreDifferences`

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
spec:
  ignoreDifferences:
    - group: apps
      kind: Deployment
      jsonPointers:
        - /spec/replicas
  syncPolicy:
    syncOptions:
      - RespectIgnoreDifferences=true
```

Per i campi aggiunti dai webhook predefiniti o mutanti di Kubernetes, puoi abilitare la differenza lato server sull'applicazione:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  annotations:
    argocd.argoproj.io/compare-options: ServerSideDiff=true,IncludeMutationWebhook=true
```

Server-side diff esegue un'applicazione a secco per risorsa, il che aumenta il carico sul server API Kubernetes. Provalo su un numero limitato di applicazioni prima di abilitarlo su larga scala.

High-churn risorse di proprietà del controller

Alcuni controller generano un gran numero di risorse di breve durata o aggiornate frequentemente. Gli esempi includono gli oggetti dei nodi Karpenter, gli oggetti identificativi ed endpoint Cilium e i report sulle policy di Kyverno. Se queste risorse generano un volume elevato di eventi di monitoraggio e causano l'interruzione della sincronizzazione, puoi ridurre il carico escludendo tali tipi di risorse o filtrando gli eventi di monitoraggio. Queste modifiche richiedono una configurazione del controller a livello di istanza.

Per quanto riguarda la funzionalità gestita, apri un caso di AWS supporto per richiedere l'esclusione delle risorse o il filtro degli eventi di controllo per questi tipi di risorse.

Best practice

- Usa prima application diff: esegui `argocd app diff` come primo passaggio diagnostico per qualsiasi problema di sincronizzazione ripetuta. Ti mostra la causa esatta della deriva.
- Preferisci le IgnoreDifferenze ristrette: scegli come target campi specifici su tipi di risorse specifici. Evita regole generiche di ignoramento che possono mascherare una reale deriva dalla configurazione.
- Associa IgnoreDifferences a RespectIgnoreDifferences: Aggiungi sempre l'`RespectIgnoreDifferences=true`opzione di sincronizzazione. Senza di essa, le sincronizzazioni continuano a sovrascrivere i campi ignorati.
- Mantieni univoci i nomi delle risorse: definisci i nomi delle risorse per ambiente e cluster per evitare conflitti di proprietà tra Applicazioni o. ApplicationSets
- Fai attenzione con prune e SelfHeal: non abilitarli entrambi su carichi di lavoro che richiedono molto tempo per essere avviati. L'autoguarigione può abbattere le risorse prima che diventino sane.

- Aggiungi i percorsi di targetRevision e scope manifest: per le applicazioni in archivi condivisi di grandi dimensioni, utilizza un ramo o un tag anziché e aggiungi l'annotazioneHEAD. `manifest-generate-paths`

Quando contattare AWS Supporto

Aprire una richiesta di AWS assistenza nelle seguenti situazioni:

- Instance-level l'ottimizzazione del controller sembra necessaria (conteggio dei processori, tempi di riparazione automatica o esclusione delle risorse).
- Repo-server oppure la capacità del controller sembra insufficiente per il numero di applicazioni.
- La configurazione, la deriva, la proprietà o i finalizzatori del carico di lavoro non spiegano il comportamento.

Includi l'output delle `argocd app get` e `argocd app diff` per le applicazioni interessate nel tuo caso di supporto.

### Fasi successive

- the section called “Considerazioni su Argo CD”- Considerazioni e best practice su Argo CD
- the section called “Lavorare con Argo CD”- Crea e gestisci le applicazioni Argo CD
- the section called “Registra i cluster”- Configurazione di implementazioni multi-cluster
- the section called “Funzionalità di risoluzione dei problemi”- Guida generale alla risoluzione dei problemi relativi alle funzionalità

## Confronto tra EKS Capability for Argo CD e Argo CD autogestito

Il CD EKS Capability for Argo offre un'esperienza Argo CD completamente gestita che viene eseguita in EKS. Per un confronto generale tra EKS Capabilities e soluzioni autogestite, vedere. the section called “Considerazioni” Questo argomento si concentra sulle CD-specific differenze di Argo, tra cui l'autenticazione, la gestione multi-cluster e il supporto delle funzionalità upstream.

### Differenze rispetto al CD Argo originale

L'EKS Capability for Argo CD si basa sul CD Argo upstream ma differisce nel modo in cui è accessibile, configurato e integrato con i servizi. AWS

RBAC e autenticazione: la funzionalità include tre ruoli RBAC (admin, editor, viewer) e utilizza AWS Identity Center per l'autenticazione anziché l'autenticazione integrata di Argo CD. Configura le mappature dei ruoli tramite il `rbacRoleMapping` parametro della funzionalità per mappare i gruppi di Identity Center ai ruoli Argo CD, non tramite i CD Argo. `argocd-rbac-cm` ConfigMap L'interfaccia utente di Argo CD è ospitata con un proprio URL diretto (lo trovi nella console EKS nella scheda Capabilities del cluster) e l'accesso all'API utilizza l' AWS autenticazione e l'autorizzazione tramite IAM.

Configurazione del cluster: la funzionalità non configura automaticamente le topologie locali di cluster o hub-and-spoke. È possibile configurare i cluster di destinazione della distribuzione e le voci di accesso EKS. La funzionalità supporta solo i cluster Amazon EKS come obiettivi di distribuzione utilizzando gli ARN del cluster EKS (non gli URL del server API Kubernetes). La funzionalità non aggiunge automaticamente il cluster locale (`kubernetes.default.svc`) come destinazione di distribuzione: per distribuirlo nello stesso cluster in cui viene creata la funzionalità, registra esplicitamente quel cluster utilizzando il relativo ARN.

Accesso remoto semplificato al cluster: la funzionalità semplifica le implementazioni multi-cluster utilizzando EKS Access Entries per concedere l'accesso Argo CD ai cluster remoti, eliminando la necessità di configurare i ruoli IAM per gli account di servizio (IRSA) o impostare presupposti di ruolo IAM tra account. La funzionalità fornisce inoltre un accesso trasparente a cluster EKS completamente privati senza richiedere peering VPC o configurazioni di rete specializzate:AWS gestisce automaticamente la connettività tra la funzionalità Argo CD e i cluster remoti privati.

Integrazione diretta dei AWS servizi: la funzionalità fornisce l'integrazione diretta con i AWS servizi tramite le autorizzazioni IAM del Capability Role. È possibile fare riferimento ai CodeCommit repository, ai grafici ECR Helm e CodeConnections direttamente nelle risorse dell'applicazione senza creare configurazioni del repository. Ciò semplifica l'autenticazione ed elimina la necessità di gestire credenziali separate per i servizi. AWS Per informazioni dettagliate, vedi the section called “Configura i repository”.

Supporto per lo spazio dei nomi: la funzionalità richiede di specificare un singolo namespace in cui è necessario creare Argo CD Application e risorse personalizzate ApplicationSet. AppProject

> **Note**
>
> Questa restrizione dello spazio dei nomi si applica solo alle risorse personalizzate di Argo CD (Application,,). ApplicationSet AppProject I carichi di lavoro delle applicazioni possono essere distribuiti in qualsiasi namespace in qualsiasi cluster di destinazione. Ad esempio, se si crea la funzionalità con namespace`argocd`, tutti i CR delle applicazioni devono essere creati nello

spazio dei nomi, ma tali applicazioni possono distribuire carichi di lavoro in,, o in qualsiasi altro `argocd` namespace. `default` `production` `staging`

> **Note**
>
> La funzionalità gestita ha requisiti specifici per l'utilizzo e la configurazione della CLI: AppProject
>
> - Quando si utilizza l'Argo CD CLI, specificate le applicazioni con il prefisso dello spazio dei nomi: `argocd app sync namespace/appname`
> - AppProject le risorse devono specificare `.spec.sourceNamespaces` per definire quali namespace il progetto può controllare per le applicazioni (in genere impostato sullo spazio dei nomi specificato durante la creazione della funzionalità)
> - Le annotazioni di tracciamento delle risorse utilizzano il formato: `namespace_appname:group/kind:namespace/name`

Funzionalità non supportate: le seguenti funzionalità non sono disponibili nella funzionalità gestita:

- Config Management Plugins (CMP) per la generazione di manifesti personalizzati
- Controller delle notifiche
- Provider SSO personalizzati (è supportato solo AWS Identity Center, inclusa l'identità federata di terze parti tramite AWS Identity Center)
- Estensioni dell'interfaccia utente e banner personalizzati
- È supportato l'accesso diretto `argocd-params` e la maggior parte delle configurazioni ConfigMaps (un sottoinsieme di `argocd-cm` campi). Vedithe section called “Configurare le impostazioni”)
- Modifica del timeout di sincronizzazione (fisso a 120 secondi)

Compatibilità: le applicazioni e il ApplicationSets funzionamento sono identici a quelli del CD Argo originale senza modifiche ai manifesti. La funzionalità utilizza le stesse API e CRD di Kubernetes, quindi strumenti come questi funzionano allo stesso modo. `kubectl` La funzionalità supporta completamente applicazioni e ApplicationSets GitOps flussi di lavoro con sincronizzazione automatica, implementazioni multi-cluster, politiche di sincronizzazione (automatizzate, prune, self-heal), onde e hook di sincronizzazione, valutazione dello stato delle risorse Kubernetes standard,

funzionalità di rollback, fonti di repository Git (HTTPS e SSH), Helm, Kustomize e semplici manifesti YAML, credenziali delle GitHub app, progetti per multi-tenancy ed esclusioni e inclusioni di risorse.

## Utilizzo dell'Argo CD CLI con la funzionalità gestita

L'Argo CD CLI funziona come l'Argo CD upstream per la maggior parte delle operazioni, ma l'autenticazione e la registrazione del cluster sono diverse.

### Prerequisiti

Installa l'Argo CD CLI seguendo le istruzioni di installazione upstream. https://argo-cd.readthedocs.io/en/stable/cli_installation/

### Configurazione

Configura la CLI utilizzando le variabili di ambiente:

1. Ottieni l'URL del server CD Argo dalla console EKS (nella scheda  Capacità del cluster) o utilizzando la CLI AWS . Il `https://` prefisso deve essere rimosso:

```
export ARGOCD_SERVER=$(aws eks describe-capability \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.configuration.argoCd.serverUrl' \
  --output text \
  --region region-code | sed 's|^https://||')
```

2. Genera un token di account dall'interfaccia utente di Argo CD (Impostazioni → Account → admin → Genera nuovo token), quindi impostalo come variabile di ambiente:

```
export ARGOCD_AUTH_TOKEN="your-token-here"
```

> **Important**
>
> Questa configurazione utilizza il token dell'account amministratore per i flussi di lavoro di configurazione e sviluppo iniziali. Per i casi d'uso in produzione, utilizza ruoli e token con ambito di progetto per seguire il principio del privilegio minimo. Per ulteriori informazioni sulla configurazione dei ruoli del progetto e dell'RBAC, vedere. the section called “Configurazione delle autorizzazioni ”

1. Impostate l'opzione gRPC richiesta:

```
export ARGOCD_OPTS="--grpc-web"
```

Con queste variabili di ambiente impostate, è possibile utilizzare l'Argo CD CLI senza il comando. `argocd login`

Differenze principali

La funzionalità gestita presenta le seguenti limitazioni CLI:

- `argocd admin`i comandi non sono supportati (richiedono l'accesso diretto al pod)
- `argocd login`non è supportato (usa invece i token dell'account o del progetto)
- `argocd cluster add`richiede il `--aws-cluster-name` flag con l'ARN del cluster EKS

Esempio: registrare un cluster

Registra un cluster EKS per la distribuzione delle applicazioni:

```
# Get the cluster ARN
CLUSTER_ARN=$(aws eks describe-cluster \
  --name my-cluster \
  --query 'cluster.arn' \
  --output text)

# Register the cluster
argocd cluster add $CLUSTER_ARN \
  --aws-cluster-name $CLUSTER_ARN \
  --name in-cluster \
  --project default
```

Per la documentazione completa della CLI di Argo CD, consultate il riferimento alla CLI di Argo CD.

## Percorso di migrazione

È possibile migrare da Argo CD autogestito alla funzionalità gestita:

1. Controlla la tua attuale configurazione del CD Argo per individuare le funzionalità non supportate (controller delle notifiche, CMP, controlli di integrità personalizzati, estensioni dell'interfaccia utente)

2. Ridimensiona i controller Argo CD autogestiti fino a zero repliche per evitare conflitti
3. Crea una risorsa con funzionalità Argo CD sul tuo cluster
4. Esporta le tue applicazioni esistenti ApplicationSets e AppProjects
5. Esegui la migrazione delle credenziali del repository, dei segreti del cluster e dei modelli di credenziali del repository (repocreds)
6. Se utilizzi chiavi GPG, certificati TLS o host noti SSH, esegui la migrazione anche di queste configurazioni
7. Aggiorna `destination.server` i campi per utilizzare i nomi dei cluster o gli ARN dei cluster EKS
8. Applicali all'istanza Argo CD gestita
9. Verifica che le applicazioni si sincronizzino correttamente
10. Disattiva l'installazione autogestita del CD Argo

La funzionalità gestita utilizza le stesse API e le stesse definizioni delle risorse di Argo CD, quindi i manifesti esistenti funzionano con modifiche minime.

### Fasi successive

- the section called "Crea una funzionalità Argo CD"- Crea una risorsa con funzionalità Argo CD
- the section called "Lavorare con Argo CD"- Implementa la tua prima applicazione
- the section called "Considerazioni su Argo CD"- Configura l'integrazione con AWS Identity Center

## Composizione delle risorse con kro (Kube Resource Orchestrator)

kro (Kube Resource Orchestrator) è un progetto open source che consente di definire API Kubernetes personalizzate utilizzando una configurazione semplice e diretta. Kubernetes-native Con kro, puoi configurare facilmente nuove API personalizzate che creano un gruppo di oggetti Kubernetes e le operazioni logiche tra di essi.

Con EKS Capabilities, kro è completamente gestito da AWS, eliminando la necessità di installare, mantenere e scalare i controller kro sui cluster.

## Come funziona kro

kro introduce una Custom Resource Definition (CRD) chiamata `ResourceGraphDefinition` (RGD) che consente la creazione semplice e semplificata di API Kubernetes personalizzate. Quando crei un`ResourceGraphDefinition`, kro utilizza le estensioni native di Kubernetes per creare e gestire nuove API nel tuo cluster. Partendo da questa specifica di singola risorsa, kro creerà e registrerà per te un nuovo CRD in base alle tue specifiche e si adatterà per gestire le tue risorse personalizzate appena definite.

Gli RGD possono includere più risorse e kro determinerà le interdipendenze e l'ordine delle risorse, quindi non è necessario. È possibile utilizzare una sintassi semplice per inserire la configurazione da una risorsa all'altra, semplificando notevolmente le composizioni ed eliminando la necessità di operatori «collanti» nel cluster. Con kro, le tue risorse personalizzate possono includere risorse Kubernetes native e tutte le Custom Resource Definitions (CRD) installate nel cluster.

kro supporta un singolo tipo di risorsa principale:

- ResourceGraphDefinition (RGD): definisce una risorsa personalizzata Kubernetes, incapsulando una o più risorse Kubernetes native o personalizzate sottostanti

Oltre a questa risorsa, kro creerà e gestirà il ciclo di vita delle risorse personalizzate create con essa, nonché tutte le risorse che le costituiscono.

kro si integra perfettamente con AWS Controllers for Kubernetes (ACK), consentendoti di comporre risorse per il carico di lavoro con risorse per creare astrazioni di livello superiore. AWS Ciò consente di creare i propri elementi costitutivi del cloud, semplificando la gestione delle risorse e abilitando modelli riutilizzabili con impostazioni di configurazione predefinite e immutabili basate sugli standard organizzativi.

## Vantaggi di kro

kro consente ai team della piattaforma di creare API Kubernetes personalizzate che compongono più risorse in astrazioni di livello superiore. Ciò semplifica la gestione delle risorse consentendo agli sviluppatori di implementare applicazioni complesse utilizzando risorse personalizzate semplici, standardizzate e con versioni diverse. Definisci modelli riutilizzabili per combinazioni di risorse comuni, consentendo una creazione coerente di risorse in tutta l'organizzazione.

kro utilizza il Common Expression Language (CEL) in Kubernetes per trasferire valori tra le risorse e incorporare la logica condizionale, offrendo flessibilità nella composizione delle risorse. Puoi

comporre sia le risorse Kubernetes che le AWS risorse gestite da ACK in API personalizzate unificate, abilitando definizioni complete di applicazioni e infrastrutture.

kro supporta la configurazione dichiarativa tramite i manifesti di Kubernetes, abilitando i GitOps flussi di lavoro e l'infrastruttura come pratiche di codice che si integrano perfettamente con i processi di sviluppo esistenti. Come parte di EKS Managed Capabilities, kro è completamente gestito da AWS, eliminando la necessità di installare, configurare e mantenere i controller kro sui cluster.

Esempio: creazione di un ResourceGraphDefinition

L'esempio seguente mostra una procedura semplice `ResourceGraphDefinition` che crea un'applicazione Web con una distribuzione e un servizio:

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: web-application
spec:
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    spec:
      name: string
      replicas: integer | default=3
  resources:
    - id: deployment
      template:
        apiVersion: apps/v1
        kind: Deployment
        metadata:
          name: ${schema.spec.name}
        spec:
          replicas: ${schema.spec.replicas}
    - id: service
      template:
        apiVersion: v1
        kind: Service
        metadata:
          name: ${schema.spec.name}
```

Quando gli utenti creano istanze della risorsa `WebApplication` personalizzata, kro crea automaticamente le risorse di distribuzione e servizio corrispondenti, gestendone il ciclo di vita insieme alla risorsa personalizzata.

## Integrazione con altre funzionalità gestite da EKS

kro si integra con altre funzionalità gestite da EKS.

- AWS Controllers for Kubernetes (ACK): usa kro per comporre le risorse ACK in astrazioni di livello superiore, semplificando la gestione delle risorse. AWS
- Argo CD: utilizza Argo CD per gestire l'implementazione di risorse personalizzate kro su più cluster, abilitando flussi di lavoro per gli elementi costitutivi della piattaforma e gli stack di applicazioni. GitOps

## Iniziare con kro

Per iniziare a usare EKS Capability for kro:

1. Crea una risorsa con funzionalità kro sul tuo cluster EKS tramite la AWS console, la AWS CLI o la tua infrastruttura preferita come strumento di codice.
2. Crea ResourceGraphDefinitions (RGD) che definiscono le tue API personalizzate e le composizioni di risorse.
3. Applica istanze delle tue risorse personalizzate per fornire e gestire Kubernetes e risorse sottostanti. AWS

## Crea una funzionalità kro

Questo argomento spiega come creare una funzionalità kro sul tuo cluster Amazon EKS.

### Prerequisiti

Prima di creare una funzionalità kro, assicurati di avere:

- Un cluster Amazon EKS esistente che esegue una versione di Kubernetes supportata (sono supportate tutte le versioni con supporto standard ed esteso)
- Autorizzazioni IAM sufficienti per creare risorse di funzionalità sui cluster EKS
- (Per CLI/eksctl) Lo strumento CLI appropriato installato e configurato

> **Note**
>
> A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive oltre alla policy di fiducia. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS Tuttavia, è comunque necessario fornire a un IAM Capability Role la policy di fiducia appropriata. Per informazioni sulla configurazione delle autorizzazioni RBAC di Kubernetes per kro, consulta. the section called "Configurazione delle autorizzazioni "

## Scegli il tuo strumento

Puoi creare una funzionalità kro usando Console di gestione AWS, AWS CLI o eksctl:

- the section called "Console"- Usa la console per un'esperienza guidata
- the section called " AWS CLI"- Usa la AWS CLI per lo scripting e l'automazione
- the section called "eksctl"- Usa eksctl per un'esperienza Kubernetes-native

## Cosa succede quando crei una funzionalità kro

Quando crei una funzionalità kro:

1. EKS crea il servizio di capacità kro e lo configura per monitorare e gestire le risorse nel cluster
2. Le definizioni delle risorse personalizzate (CRD) sono installate nel cluster
3. Viene creata automaticamente una voce di accesso per il tuo IAM Capability Role con la `AmazonEKSKROPolicy` quale concede le autorizzazioni per la gestione ResourceGraphDefinitions e le relative istanze (vedi) the section called "Considerazioni sulle funzionalità EKS"
4. La funzionalità presuppone l'IAM Capability Role fornito dall'utente (utilizzato solo per la relazione di fiducia)
5. kro inizia a controllare `ResourceGraphDefinition` le risorse e le relative istanze
6. Lo stato della capacità cambia da a `CREATING` `ACTIVE`

Una volta attivo, puoi creare o ResourceGraphDefinitions definire API personalizzate e creare istanze di tali API.

> **Note**
>
> La voce di accesso creata automaticamente include la voce `AmazonEKSKROPolicy` che concede a kro le autorizzazioni di gestione e le relative istanze. ResourceGraphDefinitions Per consentire a kro di creare le risorse Kubernetes sottostanti definite nell'utente ResourceGraphDefinitions (come le risorse Deployments, Services o ACK), è necessario configurare politiche di accesso aggiuntive. Per ulteriori informazioni sulle voci di accesso e su come configurare autorizzazioni aggiuntive, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

## Fasi successive

Dopo aver creato la funzionalità kro:

- the section called "concetti kro"- Comprendi i concetti di kro e la composizione delle risorse

## Crea una funzionalità kro utilizzando la console

Crea una funzionalità kro sul tuo cluster Amazon EKS utilizzando la console di AWS gestione.

Crea la funzionalità kro

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Capacità.
4. Nella barra di navigazione a sinistra, scegli kro (Kube Resource Orchestrator).
5. Scegli la funzionalità Create kro.
6. Per IAM Capability Role:
   - Se disponi già di un IAM Capability Role, selezionalo dal menu a discesa
   - Se devi creare un ruolo, scegli Create kro role

     Questo apre la console IAM in una nuova scheda con policy di fiducia precompilate. Il ruolo non richiede autorizzazioni IAM aggiuntive poiché kro opera interamente all'interno del cluster.

     Dopo aver creato il ruolo, torna alla console EKS e il ruolo verrà selezionato automaticamente.

> **Note**
>
> A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive oltre alla politica di fiducia. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS

7. Scegli Create (Crea).

Inizia il processo di creazione delle funzionalità.

Verifica che la funzionalità sia attiva

1. Nella scheda Funzionalità, visualizza lo stato delle capacità di kro.
2. Attendi che lo stato cambi da `CREATING` a`ACTIVE`.
3. Una volta attiva, la funzionalità è pronta per l'uso.

Per informazioni sullo stato delle funzionalità e sulla risoluzione dei problemi, vederethe section called "Lavorare con le funzionalità".

Concedi le autorizzazioni per gestire le risorse Kubernetes

Quando crei una funzionalità kro, viene creato automaticamente un EKS Access Entry con`AmazonEKSKROPolicy`, che consente a kro di gestire e le relative istanze. ResourceGraphDefinitions Tuttavia, per impostazione predefinita non viene concessa alcuna autorizzazione per creare le risorse Kubernetes sottostanti (come Deployments, Services, ecc.) definite nel tuo. ConfigMaps ResourceGraphDefinitions

Questa progettazione intenzionale segue il principio del privilegio minimo: diversi richiedono autorizzazioni diverse. ResourceGraphDefinitions Devi configurare in modo esplicito le autorizzazioni o le esigenze in base alle risorse che gestirai. ResourceGraphDefinitions

Per iniziare rapidamente, in ambienti di test o sviluppo, usa: `AmazonEKSClusterAdminPolicy`

1. Nella console EKS, accedi alla scheda Accesso del cluster.
2. In Voci di accesso, trova la voce relativa al tuo ruolo kro capability (avrà il ruolo ARN che hai creato in precedenza).
3. Scegli la voce di accesso per aprirne i dettagli.

4. Nella sezione Politiche di accesso, scegli Associa politica di accesso.
5. Seleziona `AmazonEKSClusterAdminPolicy` dall'elenco delle politiche.
6. Per l'ambito di accesso, seleziona Cluster.
7. Selezionare Associate (Associa).

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Concede ampie autorizzazioni per creare e gestire tutte le risorse Kubernetes, inclusa la possibilità di creare qualsiasi tipo di risorsa in tutti i namespace. Questa funzionalità è utile per lo sviluppo e i POC, ma non deve essere utilizzata in produzione. Per la produzione, crea politiche RBAC personalizzate che concedano solo le autorizzazioni necessarie per le risorse specifiche che gestirai. ResourceGraphDefinitions Per indicazioni sulla configurazione delle autorizzazioni con privilegi minimi, consulta e. the section called “Configurazione delle autorizzazioni ” the section called “Considerazioni sulle funzionalità EKS”

Verifica che siano disponibili risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate kro siano disponibili nel tuo cluster.

Utilizzo della console

1. Accedi al tuo cluster nella console Amazon EKS
2. Scegli la scheda Risorse
3. Scegli Estensioni
4. Scegliere CustomResourceDefinitions

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Usare kubectl

```
kubectl api-resources | grep kro.run
```

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Fasi successive

- the section called "concetti kro"- Comprendi i concetti kro e la composizione delle risorse
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa kro Capability

## Crea una funzionalità kro utilizzando la CLI AWS

Crea una funzionalità kro sul tuo cluster Amazon EKS utilizzando la AWS CLI. Questa procedura ti guida nella creazione di un ruolo IAM e nella verifica che la funzionalità sia attiva.

Prerequisiti

- AWS CLI: versione `2.12.3` o successiva. Per verificare la tua versione, `aws --version` esegui. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando.
- **`kubectl`**— Uno strumento da riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".

Fase 1: creazione di un ruolo IAM Capability

Crea un file di policy di fiducia:

```
cat > kro-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crea il ruolo IAM:

```
aws iam create-role \
  --role-name KROCapabilityRole \
  --assume-role-policy-document file://kro-trust-policy.json
```

> **Note**
>
> A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS Il ruolo è necessario solo per stabilire un rapporto di fiducia con il servizio EKS Capabilities.

Fase 2: Creare la funzionalità kro

Crea la risorsa di funzionalità kro sul tuo cluster. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster (ad esempio`us-west-2`) e *`my-cluster`* con il nome del cluster.

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --type KRO \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/KROCapabilityRole \
  --delete-propagation-policy RETAIN
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva poiché EKS crea l'infrastruttura e i componenti di funzionalità richiesti. EKS installerà le Kubernetes Custom Resource Definitions relative a questa funzionalità nel cluster non appena viene creata.

> **Note**
>
> Se ricevi un errore che indica che il cluster non esiste o non disponi delle autorizzazioni, verifica:
>
> - Il nome del cluster è corretto
> - La tua AWS CLI è configurata per la regione corretta
> - Disponi delle autorizzazioni IAM richieste

Fase 3: Verifica che la funzionalità sia attiva

Attendi che la funzionalità diventi attiva. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e sostituiscilo *my-cluster* con il nome del cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.status' \
  --output text
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE.

Puoi anche visualizzare i dettagli completi delle funzionalità:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro
```

Passaggio 4: concedere le autorizzazioni per gestire le risorse Kubernetes

Quando crei una funzionalità kro, viene creato automaticamente un EKS Access Entry conAmazonEKSKROPolicy, che consente a kro di gestire e le relative istanze. ResourceGraphDefinitions Tuttavia, per impostazione predefinita non viene concessa alcuna autorizzazione per creare le risorse Kubernetes sottostanti (come Deployments, Services, ecc.) definite nel tuo. ConfigMaps ResourceGraphDefinitions

Questa progettazione intenzionale segue il principio del privilegio minimo: diversi richiedono autorizzazioni diverse. ResourceGraphDefinitions Ad esempio: * A ResourceGraphDefinition che crea solo risorse ACK ConfigMaps e Secrets necessita di autorizzazioni diverse rispetto a una che crea distribuzioni e servizi * A ResourceGraphDefinition che crea risorse ACK necessita delle autorizzazioni per quelle risorse personalizzate specifiche * Alcuni potrebbero ResourceGraphDefinitions leggere solo le risorse esistenti senza crearne di nuove

È necessario configurare in modo esplicito le autorizzazioni o le esigenze in base alle risorse che gestirai. ResourceGraphDefinitions

Configurazione rapida

Per iniziare rapidamente, in ambienti di test o sviluppo, usa: AmazonEKSClusterAdminPolicy

Ottieni il ruolo di capacità ARN:

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associa la politica di amministrazione del cluster:

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn $CAPABILITY_ROLE_ARN \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Concede ampie autorizzazioni per creare e gestire tutte le risorse Kubernetes, inclusa la possibilità di creare qualsiasi tipo di risorsa in tutti i namespace. Questa funzionalità è utile per lo sviluppo e i POC, ma non deve essere utilizzata in produzione. Per la produzione, crea politiche RBAC personalizzate che concedano solo le autorizzazioni necessarie per le risorse specifiche che gestirai. ResourceGraphDefinitions Per indicazioni sulla configurazione delle autorizzazioni con privilegi minimi, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

Passaggio 5: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate kro siano disponibili nel tuo cluster:

```
kubectl api-resources | grep kro.run
```

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Fasi successive

- the section called "concetti kro"- Comprendi i concetti kro e la composizione delle risorse
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa kro Capability

## Crea una funzionalità kro usando eksctl

Crea una funzionalità kro sul tuo cluster Amazon EKS usando eksctl.

> **Note**
>
> I passaggi seguenti richiedono una versione eksctl o successiva. `0.215.0` Per verificare la tua versione, esegui. `eksctl version`

Fase 1: Creare un ruolo di capacità IAM

Crea un file di policy sulla fiducia:

```
cat > kro-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crea il ruolo IAM:

```
aws iam create-role \
  --role-name KROCapabilityRole \
```

```
  --assume-role-policy-document file://kro-trust-policy.json
```

> **Note**
>
> A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive oltre alla policy di trust. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS

Fase 2: Creare la funzionalità kro

Crea la funzionalità kro usando eksctl. Sostituisci *region-code* con la AWS regione in cui si trova il tuo cluster e sostituiscilo *my-cluster* con il nome del tuo cluster.

```
eksctl create capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro \
  --type KRO \
  --role-arn arn:aws:iam::111122223333:role/KROCapabilityRole
```

Il comando viene restituito immediatamente, ma la funzionalità richiede del tempo per diventare attiva.

Fase 3: Verificare che la funzionalità sia attiva

Controlla lo stato della capacità. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e sostituiscilo *my-cluster* con il nome del cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE.

Passaggio 4: concedere le autorizzazioni per gestire le risorse Kubernetes

Per impostazione predefinita, kro può creare e gestire solo le relative istanze. ResourceGraphDefinitions Per consentire a kro di creare e gestire le risorse Kubernetes

sottostanti definite nel tuo ResourceGraphDefinitions, associa la policy di accesso alla voce di `AmazonEKSClusterAdminPolicy` accesso della funzionalità.

Ottieni il ruolo di capacità ARN:

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associa la politica di amministrazione del cluster:

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn $CAPABILITY_ROLE_ARN \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Concede ampie autorizzazioni per creare e gestire tutte le risorse Kubernetes e ha lo scopo di semplificare l'avvio. Per l'uso in produzione, crea politiche RBAC più restrittive che concedano solo le autorizzazioni necessarie per le risorse specifiche che gestirai. ResourceGraphDefinitions Per indicazioni sulla configurazione delle autorizzazioni con privilegi minimi, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

Passaggio 5: verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate di kro siano disponibili nel tuo cluster:

```
kubectl api-resources | grep kro.run
```

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Fasi successive

- the section called "concetti kro"- Comprendi i concetti di Kro e la composizione delle risorse
- the section called "Lavorare con le funzionalità"- Gestisci le tue risorse di capacità kro

# concetti kro

kro consente ai team della piattaforma di creare Kubernetes personalizzati APIs che compongono più risorse in astrazioni di livello superiore. Questo argomento illustra un esempio pratico, quindi spiega i concetti fondamentali da comprendere quando si lavora con EKS Capability for kro.

## Guida introduttiva a kro

Dopo aver creato la funzionalità kro (vedithe section called "Crea una capacità kro"), puoi iniziare a crearne un APIs utilizzo personalizzato ResourceGraphDefinitions nel tuo cluster.

Ecco un esempio completo che crea una semplice astrazione di un'applicazione web:

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: webapplication
spec:
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    group: kro.run
    spec:
      name: string | required=true
      image: string | default="nginx:latest"
      replicas: integer | default=3
  resources:
  - id: deployment
    template:
      apiVersion: apps/v1
      kind: Deployment
      metadata:
        name: ${schema.spec.name}
      spec:
        replicas: ${schema.spec.replicas}
        selector:
```

```
          matchLabels:
            app: ${schema.spec.name}
        template:
          metadata:
            labels:
              app: ${schema.spec.name}
          spec:
            containers:
            - name: app
              image: ${schema.spec.image}
              ports:
              - containerPort: 80
  - id: service
    template:
      apiVersion: v1
      kind: Service
      metadata:
        name: ${schema.spec.name}
      spec:
        selector:
          app: ${schema.spec.name}
        ports:
        - protocol: TCP
          port: 80
          targetPort: 80
```

Dopo averlo applicato ResourceGraphDefinition, i team applicativi possono creare applicazioni Web utilizzando la vostra API semplificata:

```
apiVersion: kro.run/v1alpha1
kind: WebApplication
metadata:
  name: my-app
spec:
  name: my-app
  replicas: 5
```

kro crea automaticamente Deployment and Service con la configurazione appropriata. Poiché `image` non è specificato, utilizza il valore predefinito `nginx:latest` dello schema.

## Concetti principali

> **Important**
>
> kro esegue la convalida ResourceGraphDefinitions al momento della creazione, non in fase di esecuzione. Quando crei un RGD, kro convalida la sintassi CEL, controlla il tipo delle espressioni rispetto agli schemi Kubernetes effettivi, verifica l'esistenza dei campi e rileva le dipendenze circolari. Ciò significa che gli errori vengono rilevati immediatamente quando si crea l'RGD, prima che qualsiasi istanza venga distribuita.

### ResourceGraphDefinition

A ResourceGraphDefinition (RGD) definisce un'API Kubernetes personalizzata specificando:

- Schema: la struttura dell'API che utilizza il SimpleSchema formato (nomi di campo, tipi, impostazioni predefinite, convalida)
- Risorse: modelli per il Kubernetes sottostante o risorse da creare AWS
- Dipendenze: come le risorse si relazionano tra loro (rilevate automaticamente dai riferimenti di campo)

Quando applichi un RGD, kro registra una nuova Custom Resource Definition (CRD) nel tuo cluster. I team applicativi possono quindi creare istanze della vostra API personalizzata e kro si occupa della creazione e della gestione di tutte le risorse sottostanti.

Per ulteriori informazioni, consulta ResourceGraphDefinition Panoramica nella documentazione di kro.

### SimpleSchema formato

SimpleSchema fornisce un modo semplificato per definire gli schemi API senza richiedere conoscenze su OpenAPI:

```
schema:
  apiVersion: v1alpha1
  kind: Database
  spec:
    name: string | required=true description="Database name"
    size: string | default="small" enum=small,medium,large
```

```
    replicas: integer | default=1 minimum=1 maximum=5
```

SimpleSchema supportastring, integerboolean, e number tipi con vincoli comerequired,,/default, minimum e. maximum enum pattern

Per ulteriori informazioni, consulta la documentazione SimpleSchemadi kro.

espressioni CEL

kro utilizza il Common Expression Language (CEL) per fare riferimento ai valori in modo dinamico e aggiungere logica condizionale. Le espressioni CEL sono racchiuse ${ } e possono essere utilizzate in due modi:

Espressioni autonome - L'intero valore del campo è una singola espressione:

```
spec:
  replicas: ${schema.spec.replicaCount}  # Expression returns integer
  labels: ${schema.spec.labelMap}        # Expression returns object
```

Il risultato dell'espressione sostituisce l'intero valore del campo e deve corrispondere al tipo previsto del campo.

Modelli di stringhe: una o più espressioni incorporate in una stringa:

```
metadata:
  name: "${schema.spec.prefix}-${schema.spec.name}"  # Multiple expressions
  annotation: "Created by ${schema.spec.owner}"      # Single expression in string
```

Tutte le espressioni nei modelli di stringhe devono restituire stringhe. Usa string() per convertire altri tipi:"replicas-${string(schema.spec.count)}".

Riferimenti sui campi: accedi ai valori delle specifiche dell'istanza utilizzandoschema.spec:

```
template:
  metadata:
    name: ${schema.spec.name}-deployment
    namespace: ${schema.metadata.namespace}  # Can also reference metadata
  spec:
    replicas: ${schema.spec.replicas}
```

Accesso facoltativo ai campi: da utilizzare ? per campi che potrebbero non esistere:

```
# For ConfigMaps or Secrets with unknown structure
value: ${configmap.data.?DATABASE_URL}

# For optional status fields
ready: ${deployment.status.?readyReplicas > 0}
```

Se il campo non esiste, l'espressione restituisce `null` invece di fallire.

Risorse condizionali: includi le risorse solo quando sono soddisfatte le condizioni:

```
resources:
- id: ingress
  includeWhen:
    - ${schema.spec.enableIngress == true}
  template:
    # ... ingress configuration
```

Il `includeWhen` campo accetta un elenco di espressioni booleane. Tutte le condizioni devono essere vere affinché la risorsa possa essere creata. Attualmente, `includeWhen` può fare riferimento solo `schema.spec` ai campi.

Trasformazioni - Trasforma i valori utilizzando operatori e funzioni ternari:

```
template:
  spec:
    resources:
      requests:
        memory: ${schema.spec.size == "small" ? "512Mi" : "2Gi"}

    # String concatenation
    image: ${schema.spec.registry + "/" + schema.spec.imageName}

    # Type conversion
    port: ${string(schema.spec.portNumber)}
```

Riferimenti tra risorse - Valori di riferimento provenienti da altre risorse:

```
resources:
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
```

```
    kind: Bucket
    spec:
      name: ${schema.spec.name}-data

- id: configmap
  template:
    apiVersion: v1
    kind: ConfigMap
    data:
      BUCKET_NAME: ${bucket.spec.name}
      BUCKET_ARN: ${bucket.status.ackResourceMetadata.arn}
```

Quando si fa riferimento a un'altra risorsa in un'espressione CEL, si crea automaticamente una dipendenza. kro assicura che la risorsa referenziata venga creata per prima.

Per ulteriori informazioni, consulta CEL Expressions nella documentazione di kro.

Dipendenze tra le risorse

kro deduce automaticamente le dipendenze dalle espressioni CEL: non si specifica l'ordine, si descrivono le relazioni. Quando una risorsa fa riferimento a un'altra utilizzando un'espressione CEL, kro crea una dipendenza e determina l'ordine di creazione corretto.

```
resources:
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-data

- id: notification
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: BucketNotification
    spec:
      bucket: ${bucket.spec.name}  # Creates dependency: notification depends on bucket
```

L'espressione `${bucket.spec.name}` crea una dipendenza. kro crea un grafico aciclico diretto (DAG) di tutte le risorse e le relative dipendenze, quindi calcola un ordine topologico per la creazione.

Ordine di creazione: le risorse vengono create in ordine topologico (prima le dipendenze).

Creazione parallela: le risorse senza dipendenze vengono create contemporaneamente.

Ordine di eliminazione: le risorse vengono eliminate in ordine topologico inverso (prima le persone dipendenti).

Dipendenze circolari: non consentite: kro rifiuta con dipendenze circolari durante la convalida. ResourceGraphDefinitions

È possibile visualizzare l'ordine di creazione calcolato:

```
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Per maggiori informazioni, vedi Inferenza del grafico nella documentazione di kro.

## Composizione con ACK

kro funziona perfettamente con EKS Capability for ACK per comporre AWS risorse con risorse Kubernetes:

```
resources:
# Create {aws} S3 bucket with ACK
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-files

# Inject bucket details into Kubernetes ConfigMap
- id: config
  template:
    apiVersion: v1
    kind: ConfigMap
    data:
      BUCKET_NAME: ${bucket.spec.name}
      BUCKET_ARN: ${bucket.status.ackResourceMetadata.arn}

# Use ConfigMap in application deployment
- id: deployment
  template:
    apiVersion: apps/v1
    kind: Deployment
    spec:
```

```
      template:
        spec:
          containers:
          - name: app
            envFrom:
            - configMapRef:
                name: ${config.metadata.name}
```

Questo modello consente di creare AWS risorse, estrarne i dettagli (ARNs URLs, endpoint) e inserirle nella configurazione dell'applicazione, il tutto gestito come una singola unità.

Per ulteriori schemi di composizione ed esempi avanzati, consulta. the section called "Considerazioni"

### Fasi successive

- the section called "Considerazioni"- Scopri i pattern specifici di EKS, l'RBAC e l'integrazione con ACK e Argo CD
- kro Documentation - Documentazione kro completa che include espressioni CEL avanzate, modelli di convalida e risoluzione dei problemi

## Configura le autorizzazioni kro

A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM. kro opera interamente all'interno del cluster Kubernetes e non effettua chiamate API. AWS Controlla l'accesso alle risorse kro utilizzando Kubernetes RBAC standard.

### Come funzionano le autorizzazioni con kro

kro utilizza due tipi di risorse Kubernetes con ambiti diversi:

ResourceGraphDefinitions: Cluster-scoped risorse che definiscono API personalizzate. In genere gestite dai team della piattaforma che progettano e mantengono gli standard organizzativi.

Istanze: risorse Namespace-scoped personalizzate create da ResourceGraphDefinitions. Può essere creato dai team applicativi con le autorizzazioni RBAC appropriate.

Per impostazione predefinita, la funzionalità kro dispone delle autorizzazioni per gestire le relative istanze tramite ResourceGraphDefinitions la politica di accesso. `AmazonEKSKROPolicy` Tuttavia, kro richiede autorizzazioni aggiuntive per creare e gestire le risorse Kubernetes sottostanti definite nell'utente ResourceGraphDefinitions (come Deployments, Services o risorse ACK). È necessario

concedere queste autorizzazioni tramite le politiche di accesso o Kubernetes RBAC. Per i dettagli sulla concessione di queste autorizzazioni, consulta, o. the section called "eksctl" the section called " AWS CLI" the section called "Console"

## Autorizzazioni del team della piattaforma

I team della piattaforma necessitano delle autorizzazioni per creare e gestire. ResourceGraphDefinitions

Esempio ClusterRole per i team della piattaforma:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-platform-admin
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["*"]
```

Associati ai membri del team della piattaforma:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: platform-team-kro-admin
subjects:
- kind: Group
  name: platform-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-platform-admin
  apiGroup: rbac.authorization.k8s.io
```

## Autorizzazioni del team dell'applicazione

I team applicativi necessitano delle autorizzazioni per creare istanze di risorse personalizzate nei propri namespace.

Esempio di ruolo per i team delle applicazioni:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kro-app-developer
  namespace: my-app
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps", "databases"]
  verbs: ["create", "get", "list", "update", "delete", "patch"]
```

Associarsi ai membri del team applicativo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-kro-developer
  namespace: my-app
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: kro-app-developer
  apiGroup: rbac.authorization.k8s.io
```

> **Note**
>
> Il gruppo di API nel ruolo (`kro.run`in questo esempio) deve corrispondere a quello `apiVersion` definito nello schema ResourceGraphDefinition dell'utente.

## Read-only accesso

Concedi l'accesso in sola lettura alla visualizzazione ResourceGraphDefinitions e alle istanze senza autorizzazioni di modifica.

Read-only ClusterRole:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  name: kro-viewer
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["get", "list", "watch"]
```

Read-only Ruolo per le istanze:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kro-instance-viewer
  namespace: my-app
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps", "databases"]
  verbs: ["get", "list", "watch"]
```

## Multi-namespace accesso

Concedi ai team delle applicazioni l'accesso a più namespace utilizzando with. ClusterRoles RoleBindings

ClusterRole per l'accesso a più namespace:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-multi-namespace-developer
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps"]
  verbs: ["create", "get", "list", "update", "delete"]
```

Associa a namespace specifici:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-dev-access
  namespace: development
```

```
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-multi-namespace-developer
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-staging-access
  namespace: staging
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-multi-namespace-developer
  apiGroup: rbac.authorization.k8s.io
```

## Best practice

Principio del privilegio minimo: concedi solo le autorizzazioni minime necessarie per le responsabilità di ogni team.

Usa i gruppi anziché i singoli utenti: associa i ruoli ai gruppi anziché ai singoli utenti per una gestione più semplice.

Aspetti distinti relativi alla piattaforma e all'applicazione: i team della piattaforma gestiscono le istanze ResourceGraphDefinitions, i team delle applicazioni gestiscono le istanze.

Isolamento dello spazio dei nomi: utilizza i namespace per isolare team o ambienti diversi, con RBAC che controlla l'accesso a ciascun namespace.

Read-only accesso per il controllo: Fornisci l'accesso in sola lettura ai team di sicurezza e conformità per scopi di controllo.

## Fasi successive

- the section called "concetti kro"- Comprendi i nostri concetti e la composizione delle risorse

- the section called "Considerazioni sulle funzionalità EKS"- Esamina le migliori pratiche di sicurezza per quanto riguarda le funzionalità

## considerazioni kro per EKS

Esamina le considerazioni importanti sull'utilizzo della funzionalità gestita EKS per kro, tra cui quando utilizzare la composizione delle risorse, i modelli RBAC e l'integrazione con altre funzionalità EKS.

### Quando usare kro

kro è progettato per creare modelli di infrastruttura riutilizzabili e API personalizzate che semplificano la gestione complessa delle risorse.

Usa kro quando devi:

- Crea piattaforme self-service con API semplificate per i team applicativi
- Standardizza i modelli di infrastruttura tra i team (database, backup e monitoraggio)
- Gestisci le dipendenze delle risorse e trasferisci i valori tra le risorse
- Crea astrazioni personalizzate che nascondono la complessità dell'implementazione
- Componi più risorse ACK in blocchi costitutivi di livello superiore
- Abilita i GitOps flussi di lavoro per stack di infrastrutture complessi

Non usare kro quando:

- Gestione di risorse semplici e autonome (usa direttamente le risorse ACK o Kubernetes)
- È necessaria una logica di runtime dinamica (kro è dichiarativo, non imperativo)
- Le risorse non hanno dipendenze o configurazioni condivise

kro eccelle nella creazione di «percorsi asfaltati», schemi ponderati e riutilizzabili che consentono ai team di implementare correttamente infrastrutture complesse.

### Modelli RBAC

kro consente la separazione delle preoccupazioni tra i team della piattaforma che creano le istanze ResourceGraphDefinitions e i team delle applicazioni che creano le istanze.

Responsabilità del team di piattaforma

I team della piattaforma creano e gestiscono ResourceGraphDefinitions (RGD) che definiscono API personalizzate.

Autorizzazioni necessarie:

- Crea, aggiorna, elimina ResourceGraphDefinitions
- Gestisci i tipi di risorse sottostanti (implementazioni, servizi, risorse ACK)
- Accesso a tutti i namespace in cui verranno utilizzati gli RGD

Esempio ClusterRole per il team della piattaforma:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: platform-kro-admin
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["*"]
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["get", "list", "watch"]
```

Per una configurazione RBAC dettagliata, vedere. the section called "Configurazione delle autorizzazioni "

Responsabilità del team di applicazione

I team applicativi creano istanze di risorse personalizzate definite da RGD senza dover comprendere la complessità sottostante.

Autorizzazioni necessarie:

- Creare, aggiornare ed eliminare istanze di risorse personalizzate
- Accesso in lettura al loro namespace
- Nessun accesso alle risorse sottostanti o agli RGD

Vantaggi:

- I team utilizzano API semplici e di alto livello
- I team della piattaforma controllano i dettagli dell'implementazione
- Riduzione del rischio di errori di configurazione
- Onboarding più rapido per i nuovi membri del team

## Integrazione con altre funzionalità EKS

### Composizione di risorse ACK

kro è particolarmente potente se combinato con ACK per creare modelli di infrastruttura.

Schemi comuni:

- Applicazione con archiviazione: bucket S3+coda SQS+configurazione delle notifiche
- Stack di database: istanza RDS + gruppo di parametri + gruppo di sicurezza + segreto Secrets Manager
- Rete: VPC + sottoreti + tabelle di routing + gruppi di sicurezza + gateway NAT
- Elaborazione con storage: istanza EC2 + volumi EBS + profilo di istanza IAM

kro gestisce l'ordine delle dipendenze, passa i valori tra le risorse (come ARN e stringhe di connessione) e gestisce l'intero ciclo di vita come una singola unità.

Per esempi di composizione di risorse ACK, vedi. the section called “Concetti ACK”

### GitOps con Argo CD

Usa il CD EKS Capability for Argo per distribuire sia RGD che istanze dai repository Git.

Organizzazione del repository:

- Repo della piattaforma: contiene contenuti ResourceGraphDefinitions gestiti dal team della piattaforma
- Repo di applicazioni: contengono istanze di risorse personalizzate gestite dai team delle app
- Repo condiviso: contiene sia RGD che istanze per organizzazioni più piccole

Considerazioni:

- Implementa gli RGD prima delle istanze (le onde di sincronizzazione di Argo CD possono aiutarti)
- Utilizzate progetti Argo CD separati per i team di piattaforme e applicazioni
- Il team della piattaforma controlla l'accesso al repository RGD
- I team applicativi hanno accesso in sola lettura alle definizioni RGD

Per ulteriori informazioni su Argo CD, vedere. the section called "Lavorare con Argo CD"

## Organizzare ResourceGraphDefinitions

Organizza gli RGD per scopo, complessità e proprietà.

Per scopo:

- Infrastruttura: stack di database, rete, archiviazione
- Applicazione: app Web, API, processi in batch
- Piattaforma: servizi condivisi, monitoraggio, registrazione

Per complessità:

- Semplice: 2-3 risorse con dipendenze minime
- Moderato: 5-10 risorse con alcune dipendenze
- Complesso: più di 10 risorse con dipendenze complesse

Convenzioni di denominazione:

- Usa nomi descrittivi:, `webapp-with-database` `s3-notification-queue`
- Includi la versione nel nome per annullare le modifiche: `webapp-v2`
- Usa prefissi coerenti per gli RGD correlati:, `platform-` `app-`

Strategia del namespace:

- Gli RGD sono con ambito cluster (non con namespace)
- Le istanze hanno un namespace
- Usa i selettori di namespace negli RGD per controllare dove è possibile creare le istanze

## Controllo delle versioni e aggiornamenti

Piano per l'evoluzione di RGD e la migrazione delle istanze.

Aggiornamenti RGD:

- Non-breaking modifiche: aggiorna RGD in atto (aggiungi campi opzionali, nuove risorse con IncludeWhen)
- Ultime modifiche: crea un nuovo RGD con un nome diverso (webapp-v2)
- Deprecazione: contrassegna i vecchi RGD con annotazioni, comunica la cronologia della migrazione

Migrazione delle istanze:

- Crea nuove istanze con RGD aggiornato
- Convalida il corretto funzionamento delle nuove istanze
- Elimina le vecchie istanze
- kro gestisce automaticamente gli aggiornamenti delle risorse sottostanti

Migliori pratiche:

- Verifica innanzitutto le modifiche RGD in ambienti non di produzione
- Usa il controllo semantico delle versioni nei nomi RGD per le modifiche principali
- Modifiche di interruzione dei documenti e percorsi di migrazione
- Fornisci esempi di migrazione per i team applicativi

## Convalida e test

Convalida gli RGD prima di passare alla produzione.

Strategie di convalida:

- Convalida dello schema: kro convalida automaticamente la struttura RGD
- Dry-run istanze: crea istanze di test nei namespace di sviluppo
- Test di integrazione: verifica che le risorse composte funzionino insieme

- Applicazione delle politiche: utilizza i controller di ammissione per far rispettare gli standard organizzativi

Problemi comuni da testare:

- Dipendenze e ordinamento delle risorse
- Scambio di valore tra risorse (espressioni CEL)
- Inclusione condizionale delle risorse (includeWhen)
- Propagazione dello stato dalle risorse sottostanti
- Autorizzazioni RBAC, ad esempio la creazione

## Documentazione upstream

Per informazioni dettagliate sull'uso di kro:

- Guida introduttiva a kro - Creazione ResourceGraphDefinitions
- Espressioni CEL - Scrittura di espressioni CEL
- kro Guides - Modelli di composizione avanzati
- Risoluzione dei problemi - Risoluzione dei problemi e debug

## Fasi successive

- the section called "Configurazione delle autorizzazioni "- Configura RBAC per i team di piattaforme e applicazioni
- the section called "concetti kro"- Comprendi i concetti kro e il ciclo di vita delle risorse
- the section called "Risoluzione dei problemi"- Risolvi i problemi di kro
- the section called "Concetti ACK"- Scopri le risorse ACK per la composizione
- the section called "Lavorare con Argo CD"- Distribuisci RGD e istanze con GitOps

# Risolvi i problemi relativi alle funzionalità kro

> **Note**
>
> Le funzionalità EKS sono completamente gestite ed eseguite all'esterno del cluster. Non hai accesso diretto ai namespace dei controller. È possibile configurare la distribuzione dei log del controller per la visibilità del comportamento del controller. Consulta the section called "Funzionalità, registri dei controller". La risoluzione dei problemi si concentra sullo stato delle funzionalità, sulla configurazione RBAC e sullo stato delle risorse.

## La funzionalità è ATTIVA ma non funziona ResourceGraphDefinitions

Se la tua capacità kro mostra `ACTIVE` lo stato ma ResourceGraphDefinitions non sta creando risorse sottostanti, controlla lo stato della funzionalità, le autorizzazioni RBAC e lo stato delle risorse.

Verifica lo stato della funzionalità:

È possibile visualizzare i problemi relativi all'integrità e allo stato delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la  scheda  Funzionalità per visualizzare lo stato e lo stato di tutte le funzionalità.

AWS CLI:

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro

# Look for issues in the health section
```

Cause comuni:

- Autorizzazioni RBAC mancanti: kro non dispone delle autorizzazioni per creare le risorse Kubernetes sottostanti
- Espressioni CEL non valide: errori di sintassi in ResourceGraphDefinition
- Dipendenze tra le risorse: risorse dipendenti non pronte
- Convalida dello schema: l'istanza non soddisfa i requisiti dello schema RGD

Verifica le autorizzazioni RBAC:

```
# Check if capability has cluster admin policy
kubectl get accessentry -A | grep kro
```

Se la funzionalità non dispone delle autorizzazioni richieste, associale alla voce di accesso della `AmazonEKSClusterAdminPolicy` funzionalità kro o crea politiche RBAC più restrittive per l'uso in produzione. Per informazioni dettagliate, vedi the section called "Configurazione delle autorizzazioni ".

Verifica lo stato: ResourceGraphDefinition

```
# List all RGDs
kubectl get resourcegraphdefinition

# Describe specific RGD
kubectl describe resourcegraphdefinition my-rgd

# Check for validation errors
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.conditions}'
```

ResourceGraphDefinitions hanno tre condizioni di stato chiave:

- `ResourceGraphAccepted`- Se l'RGD ha superato la convalida (sintassi CEL, controllo del tipo, esistenza del campo)
- `KindReady`- Se il CRD per la tua API personalizzata è stato generato e registrato
- `ControllerReady`- Se kro sta controllando attivamente le istanze della tua API personalizzata

In caso `ResourceGraphAccepted` `False` affermativo, controlla il messaggio di condizione per individuare eventuali errori di convalida, ad esempio campi sconosciuti, mancata corrispondenza tra i tipi o dipendenze circolari.

## Istanze create ma le risorse sottostanti non vengono visualizzate

Se esistono istanze di risorse personalizzate ma le risorse Kubernetes sottostanti (Deployments, Services ConfigMaps) non vengono create, verifica che kro disponga delle autorizzazioni e verifica la presenza di errori di composizione.

Controlla lo stato dell'istanza:

```
# Describe the instance (replace with your custom resource kind and name)
kubectl describe custom-kind my-instance

# View instance events
kubectl get events --field-selector involvedObject.name=my-instance

# Check instance status conditions
kubectl get custom-kind my-instance -o jsonpath='{.status.conditions}'

# Check instance state
kubectl get custom-kind my-instance -o jsonpath='{.status.state}'
```

Le istanze hanno un `state` campo che mostra lo stato di alto livello:

- `ACTIVE`- L'istanza è in esecuzione correttamente
- `IN_PROGRESS`- L'istanza è in fase di elaborazione o riconciliazione
- `FAILED`- La riconciliazione dell'istanza non è riuscita
- `DELETING`- L'istanza viene eliminata
- `ERROR`- Si è verificato un errore durante l'elaborazione

Le istanze hanno anche quattro condizioni di stato:

- `InstanceManaged`- I finalizzatori e le etichette sono impostati correttamente
- `GraphResolved`- Creazione del grafico di runtime e risoluzione delle risorse
- `ResourcesReady`- Tutte le risorse create e pronte
- `Ready`- Lo stato generale dell'istanza (lo diventa solo `True` quando tutte le condizioni secondarie lo sono`True`)

Concentrati sulla `Ready` condizione per determinare lo stato dell'istanza. In caso `Ready` `False` affermativo, controlla le condizioni secondarie per identificare quale fase non è riuscita.

Verifica le autorizzazioni RBAC:

La funzionalità kro necessita delle autorizzazioni per creare le risorse Kubernetes sottostanti definite nel tuo. ResourceGraphDefinitions

```
# Check if the capability has the AmazonEKSClusterAdminPolicy
kubectl get accessentry -A | grep kro
```

Se mancano le autorizzazioni, associale alla voce di accesso della `AmazonEKSClusterAdminPolicy` funzionalità kro o crea politiche RBAC più restrittive per l'uso in produzione. Per informazioni dettagliate, vedi the section called "Configurazione delle autorizzazioni ".

## Errori di espressione CEL

Gli errori di espressione CEL vengono rilevati al momento ResourceGraphDefinition della creazione, non quando vengono create le istanze. kro convalida tutta la sintassi CEL, verifica i tipi delle espressioni rispetto agli schemi Kubernetes e verifica l'esistenza dei campi quando si crea l'RGD.

Errori di convalida CEL comuni:

- Riferimento al campo non definito: riferimento a un campo che non esiste nello schema o nella risorsa
- Tipo non corrispondente: l'espressione restituisce un tipo errato (ad esempio, una stringa in cui è previsto un numero intero)
- Sintassi non valida: parentesi, virgolette o operatori mancanti nell'espressione CEL
- Tipo di risorsa sconosciuto: riferimento a un CRD che non esiste nel cluster

Verifica lo stato di convalida RGD:

```
# Check if RGD was accepted
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.conditions[?
(@.type=="ResourceGraphAccepted")]}'

# View detailed validation errors
kubectl describe resourcegraphdefinition my-rgd
```

In caso `ResourceGraphAccepted` `False` affermativo, il messaggio di condizione contiene l'errore di convalida.

Esempio di espressioni CEL valide:

```
# Reference schema field
${schema.spec.appName}

# Conditional expression
${schema.spec.replicas > 1}

# String template (expressions must return strings)
name: "${schema.spec.appName}-service"

# Standalone expression (can be any type)
replicas: ${schema.spec.replicaCount}

# Resource reference
${deployment.status.availableReplicas}

# Optional field access (returns null if field doesn't exist)
${configmap.data.?DATABASE_URL}
```

## Le dipendenze delle risorse non si risolvono

kro deduce automaticamente le dipendenze dalle espressioni CEL e crea le risorse nell'ordine corretto. Se le risorse non vengono create come previsto, controlla l'ordine delle dipendenze e la disponibilità delle risorse.

Visualizza l'ordine di creazione calcolato:

```
# See the order kro will create resources
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Questo mostra l'ordine calcolato in base ai riferimenti alle espressioni CEL tra le risorse.

Verifica la disponibilità delle risorse:

```
# View instance status to see which resources are ready
kubectl get custom-kind my-instance -o jsonpath='{.status}'

# Check specific resource status
kubectl get deployment my-deployment -o jsonpath='{.status.conditions}'
```

Verifica le condizioni ReadyWhen (se utilizzate):

Il campo `readyWhen` è facoltativo. Se non specificate, le risorse sono considerate pronte subito dopo la creazione. Se hai definito `readyWhen` delle condizioni, verifica che controllino correttamente la disponibilità delle risorse:

```
resources:
  - id: deployment
    readyWhen:
      - ${deployment.status.availableReplicas == deployment.spec.replicas}
```

Controlla gli eventi  delle risorse:

```
# View events for the underlying resources
kubectl get events -n namespace --sort-by='.lastTimestamp'
```

## Errori di convalida dello schema

Se le istanze non vengono create a causa di errori di convalida dello schema, verificate che l'istanza soddisfi i requisiti dello schema RGD.

Controlla gli errori di convalida:

```
# Attempt to create instance and view error
kubectl apply -f instance.yaml

# View existing instance validation status
kubectl describe custom-kind my-instance | grep -A 5 "Validation"
```

Problemi di convalida comuni:

- Campi obbligatori mancanti: l'istanza non fornisce tutti i campi dello schema obbligatori
- Mancata corrispondenza del tipo: fornisce una stringa in cui è previsto un numero intero
- Valore enum non valido: utilizzo di un valore non presente nell'elenco consentito
- Mancata corrispondenza del modello: la stringa non corrisponde al modello regex

Rivedi lo schema RGD:

```
# View the schema definition
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.spec.schema}'
```

Assicurati che la tua istanza fornisca tutti i campi obbligatori con i tipi corretti.

### Fasi successive

- the section called "Considerazioni"- considerazioni e migliori pratiche
- the section called "Configurazione delle autorizzazioni "- Configura RBAC per i team di piattaforma e applicazione
- the section called "concetti kro"- Comprendi i concetti di Kro e il ciclo di vita delle risorse
- the section called "Funzionalità di risoluzione dei problemi"- Guida generale alla risoluzione dei problemi relativi alle funzionalità

## Confronto tra EKS Capability for kro e kro autogestito

EKS Capability for kro offre le stesse funzionalità di kro autogestito, ma con vantaggi operativi significativi. Per un confronto generale tra EKS Capabilities e soluzioni autogestite, vedi. the section called "Considerazioni"

EKS Capability for kro utilizza gli stessi controller kro upstream ed è completamente compatibile con upstream kro. ResourceGraphDefinitions, Le espressioni CEL e la composizione delle risorse funzionano in modo identico. Per la documentazione completa e gli esempi di kro, consulta la documentazione di kro.

### Percorso di migrazione

È possibile migrare da kro autogestito a funzionalità gestita senza tempi di inattività.

> **Important**
>
> Prima della migrazione, assicurati che il tuo controller kro autogestito utilizzi la stessa versione di EKS Capability for kro. Verifica la versione con funzionalità nella console EKS o in uso`aws eks describe-capability`, quindi aggiorna la tua installazione autogestita in modo che corrisponda. In questo modo si evitano problemi di compatibilità durante la migrazione.

1. Aggiorna il tuo controller kro autogestito da utilizzare per i contratti di locazione elettorali `kube-system` per i leader:

```
helm upgrade --install kro \
  oci://ghcr.io/awslabs/kro/kro-chart \
  --namespace kro \
  --set leaderElection.namespace=kube-system
```

Questo sposta il contratto di locazione del controllore`kube-system`, permettendo alla capacità gestita di coordinarsi con esso.

2. Crea la funzionalità kro sul tuo cluster (vedi) the section called "Crea una capacità kro"
3. La funzionalità gestita riconosce le istanze esistenti ResourceGraphDefinitions e prende il sopravvento sulla riconciliazione
4. Ridimensiona gradualmente o rimuovi le implementazioni kro autogestite:

```
helm uninstall kro --namespace kro
```

Questo approccio consente a entrambi i controller di coesistere in sicurezza durante la migrazione. La funzionalità gestita adotta ResourceGraphDefinitions automaticamente istanze precedentemente gestite da kro autogestito, garantendo una riconciliazione continua senza conflitti.

### Fasi successive

- the section called "Crea una capacità kro"- Crea una risorsa con funzionalità kro
- the section called "concetti kro"- Comprendi i concetti kro e la composizione delle risorse

## Risoluzione dei problemi relativi alle funzionalità EKS

> **Note**
>
> Le funzionalità EKS sono completamente gestite ed eseguite all'esterno del cluster. Non hai accesso diretto ai namespace dei controller. La risoluzione dei problemi si concentra sullo stato delle funzionalità, sullo stato delle risorse, sulla configurazione e sui log dei controller. È possibile configurare la distribuzione dei log del controller per ottenere visibilità sul comportamento del controller. Consulta the section called "Funzionalità, registri dei controller".

## Approccio generale alla risoluzione dei problemi

Per la risoluzione dei problemi di EKS Capabilities, seguite questo approccio generale:

1. Verifica dello stato delle funzionalità: utilizzalo `aws eks describe-capability` per visualizzare lo stato della funzionalità e i problemi di integrità
2. Verifica lo stato delle risorse: controlla le risorse Kubernetes (CRD) che hai creato per verificare le condizioni di stato e gli eventi
3. Esamina i registri del controller: se la consegna dei log è configurata, interroga il controller per individuare eventuali errori e dettagli di riconciliazione
4. Rivedi le autorizzazioni IAM: assicurati che il Capability Role disponga delle autorizzazioni necessarie
5. Verifica la configurazione: verifica che la configurazione specifica della funzionalità sia corretta

## Usa i log del controller per la risoluzione dei problemi

Se hai configurato la distribuzione dei log dei controller (vedithe section called "Funzionalità, registri dei controller"), puoi interrogare i log per identificare errori di riconciliazione, conflitti di risorse e problemi di configurazione.

### Errori di interrogazione su tutti i controller

```
fields @timestamp, controller, message, error
| filter level = "error"
| sort @timestamp desc
| limit 50
```

### Filtra i log per uno specifico controller di servizio ACK

Usa il `controllerGroup` campo per isolare i log da uno specifico controller di servizio ACK:

```
fields @timestamp, message, error
| filter controllerGroup = "s3.services.k8s.aws"
| filter level = "error"
| sort @timestamp desc
```

Per filtrare ulteriormente per tipo di risorsa (ad esempio, solo `SecurityGroup` i log del controller EC2):

```
fields @timestamp, message, error
| filter controllerGroup = "ec2.services.k8s.aws"
| filter controllerKind = "SecurityGroup"
| sort @timestamp desc
| limit 100
```

### Filtrare i log per una specifica applicazione Argo CD

Usa il `application` campo per isolare i log per una particolare applicazione Argo CD:

```
fields @timestamp, message, error
| filter application = "my-application"
| sort @timestamp desc
| limit 100
```

### Tieni traccia della riconciliazione per una risorsa specifica

Usa il `reconcileID` campo per seguire un singolo ciclo di riconciliazione:

```
fields @timestamp, level, message, error
| filter reconcileID = "your-reconcile-id"
| sort @timestamp asc
```

### Schemi di registro comuni che indicano problemi

- Errori di riconciliazione ripetuti: il controller non è in grado di raggiungere lo stato desiderato per una risorsa. Controlla il `error` campo per dettagli come errori di autorizzazione IAM o configurazioni di risorse non valide.
- «Errore di riconciliazione» con errori AWS API: al Capability Role potrebbero mancare le autorizzazioni per l'operazione specifica del servizio. AWS Rivedi il messaggio di errore e aggiorna le policy IAM di conseguenza.
- Nessuna voce di registro per una risorsa: se non vedi i log di una risorsa che prevedi che il controller si riconcili, verifica che la funzionalità sia presente `ACTIVE` e che la risorsa esista in uno spazio dei nomi a cui la funzionalità può accedere.

## Verifica lo stato della funzionalità

Tutte le funzionalità EKS forniscono informazioni sullo stato tramite la console EKS e l'`describe-capability`API.

Console:

1. Apri la console Amazon EKS all'indirizzo https://console.aws.amazon.com/eks/home #/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la  scheda  Funzionalità per visualizzare lo stato e lo stato di tutte le funzionalità.

La scheda Funzionalità mostra:

- Nome e tipo di funzionalità
- Stato attuale
- Problemi di salute, con descrizione

AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name
```

La risposta include:

- stato: stato attuale della capacità (`CREATING`,`ACTIVE`,`UPDATING`, `DELETINGCREATE_FAILED`,`UPDATE_FAILED`)
- stato di salute: informazioni sanitarie, inclusi eventuali problemi rilevati dalla funzionalità

## Stati di capacità comuni

CREAZIONE: la funzionalità è in fase di configurazione.

ATTIVO: La capacità è attiva e pronta all'uso. Se le risorse non funzionano come previsto, controlla lo stato delle risorse e le autorizzazioni IAM.

AGGIORNAMENTO: le modifiche alla configurazione sono in corso di applicazione. Attendi il ripristino dello stato`ACTIVE`.

CREATE_FAILED o UPDATE_FAILED: durante l'installazione o l'aggiornamento si è verificato un errore. Per i dettagli, consulta la sezione sullo stato di salute. Cause comuni:

- Politica di fiducia del ruolo IAM errata o mancante
- Il ruolo IAM non esiste o non è accessibile
- Problemi di accesso al cluster
- Parametri di configurazione non validi

## Verifica lo stato delle risorse Kubernetes

EKS Capabilities crea e gestisce le Kubernetes Custom Resource Definitions (CRD) nel tuo cluster. Durante la risoluzione dei problemi, controlla lo stato delle risorse che hai creato:

```
# List resources of a specific type
kubectl get resource-kind -A

# Describe a specific resource to see conditions and events
kubectl describe resource-kind resource-name -n namespace

# View resource status conditions
kubectl get resource-kind resource-name -n namespace -o jsonpath='{.status.conditions}'

# View events related to the resource
kubectl get events --field-selector involvedObject.name=resource-name -n namespace
```

Le condizioni relative allo stato delle risorse forniscono informazioni su:

- Se la risorsa è pronta
- Eventuali errori riscontrati
- Stato attuale della riconciliazione

# Rivedi le autorizzazioni IAM e l'accesso al cluster

Molti problemi di funzionalità derivano da problemi di autorizzazione IAM o da una configurazione di accesso al cluster mancante. Verifica sia le autorizzazioni del Capability Role che le voci di accesso al cluster.

## Controlla le autorizzazioni del ruolo IAM

Verifica che il Capability Role disponga delle autorizzazioni necessarie:

```
# List attached managed policies
aws iam list-attached-role-policies --role-name my-capability-role

# List inline policies
aws iam list-role-policies --role-name my-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-capability-role --policy-name policy-name

# View the role's trust policy
aws iam get-role --role-name my-capability-role --query 'Role.AssumeRolePolicyDocument'
```

La politica di fiducia deve consentire al responsabile del `capabilities.eks.amazonaws.com` servizio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

## Controlla le voci di accesso e le politiche di accesso di EKS

Tutte le funzionalità richiedono le voci di accesso e le politiche di accesso EKS appropriate sul cluster in cui operano.

Verifica l'esistenza di una voce di accesso:

```
aws eks list-access-entries \
  --cluster-name my-cluster \
  --region region-code
```

Cerca il Capability Role ARN nell'elenco. Se manca, la funzionalità non può accedere al cluster.

Controlla le politiche di accesso allegate alla voce:

```
aws eks list-associated-access-policies \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::111122223333:role/my-capability-role \
  --region region-code
```

Tutte le funzionalità richiedono politiche di accesso appropriate:

- ACK: necessita delle autorizzazioni per creare e gestire le risorse Kubernetes
- kro: Necessita delle autorizzazioni per creare e gestire le risorse Kubernetes
- Argo CD: necessita delle autorizzazioni per creare e gestire le applicazioni e richiede le voci di accesso su cluster di destinazione remoti per implementazioni multi-cluster

Per le implementazioni multi-cluster di Argo CD:

In caso di distribuzione su cluster remoti, verifica che il Capability Role abbia una voce di accesso su ogni cluster di destinazione:

```
# Check Access Entry on target cluster
aws eks describe-access-entry \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::111122223333:role/argocd-capability-role \
  --region region-code
```

Se l'Access Entry non è presente in un cluster di destinazione, Argo CD non può distribuire applicazioni su di esso. the section called “Registra i cluster”Per i dettagli di configurazione, vedere.

# Capability-specific risoluzione dei problemi

Per una guida dettagliata alla risoluzione dei problemi specifica per ogni tipo di funzionalità:

- the section called "Risoluzione dei problemi"- Risolvi i problemi relativi alla creazione di risorse ACK, alle autorizzazioni IAM e all'accesso tra account
- the section called "Risoluzione dei problemi"- Risolvi i problemi di sincronizzazione delle applicazioni, autenticazione del repository e implementazioni multi-cluster
- the section called "Risoluzione dei problemi"- Risoluzione dei problemi ResourceGraphDefinitions, delle espressioni CEL e delle autorizzazioni RBAC

# Problemi comuni a tutte le funzionalità

## Capacità bloccata nello stato CREATING

Se una capacità rimane nello `CREATING` stato più a lungo del previsto:

1. Verifica lo stato delle funzionalità per problemi specifici nella console (Observability > Monitor cluster > scheda Capabilities) o utilizzando la AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name \
  --query 'capability.health'
```

2. Verifica che il ruolo IAM esista e abbia la politica di fiducia corretta
3. Assicurati che il tuo cluster sia accessibile e integro
4. Verifica eventuali problemi a livello di cluster che potrebbero impedire la configurazione delle funzionalità

## Risorse non create o aggiornate

Se la funzionalità è `ACTIVE` attiva ma le risorse non vengono create o aggiornate:

1. Controlla lo stato delle risorse per verificare la presenza di eventuali condizioni di errore
2. Verifica le autorizzazioni IAM per i AWS servizi (ACK) o i repository specifici (Argo CD)

3. Controlla le autorizzazioni RBAC per la creazione delle risorse sottostanti (kro)
4. Rivedi le specifiche delle risorse per individuare eventuali errori di convalida

### L'integrità delle capacità mostra problemi

Se `describe-capability` mostra problemi di salute:

1. Leggi attentamente le descrizioni dei problemi: spesso indicano il problema specifico
2. Risolvi la causa principale (autorizzazioni IAM, errori di configurazione, ecc.)
3. La funzionalità verrà ripristinata automaticamente una volta risolto il problema

## Fasi successive

- the section called "Lavorare con le funzionalità"- Gestisci le risorse relative alle capacità
- the section called "Risoluzione dei problemi"- ACK-specific risoluzione dei problemi
- the section called "Risoluzione dei problemi"- risoluzione dei problemi relativi a Argo CD-specific
- the section called "Risoluzione dei problemi"- risoluzione dei problemi specifici per kro
- the section called "Considerazioni sulle funzionalità EKS"- Le migliori pratiche di sicurezza per le funzionalità

# Organizzazione e monitoraggio delle risorse del cluster

Questo capitolo include i seguenti argomenti per aiutarti a gestire il cluster. Puoi anche visualizzare informazioni sulle tue risorse Kubernetes con. Console di gestione AWS

- La dashboard Kubernetes è un'interfaccia utente generica basata sul web per i cluster Kubernetes. Consente agli utenti di gestire le applicazioni in esecuzione nel cluster e risolverne i problemi, nonché di gestire il cluster stesso. Per ulteriori informazioni, consulta il repository Kubernetes Dashboard. GitHub
- the section called "Metrics Server" – Il server dei parametri Kubernetes è un aggregatore dei dati di utilizzo delle risorse nel cluster. Non viene implementato per impostazione predefinita nel cluster, ma viene utilizzato dai componenti aggiuntivi di Kubernetes, ad esempio la dashboard Kubernetes e the section called "Horizontal Pod Autoscaler". In questo argomento viene descritto come installare il server dei parametri.
- the section called "Implementazione di app con Helm" – Il programma di gestione del pacchetto Helm per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes. Questo argomento consente di installare ed eseguire i file binari Helm per consentire l'installazione e la gestione di grafici utilizzando la CLI Helm nel computer.
- the section called "Tagging delle risorse ": per semplificare la gestione delle risorse Amazon EKS, è possibile assegnare metadati personalizzati a ciascuna risorsa sotto forma di tag. Questo argomento descrive i tag e mostra come crearli.
- the section called "Service Quotas"— Il tuo AWS account ha delle quote predefinite, precedentemente denominate limiti, per ogni servizio. AWS Scopri le quote per Amazon EKS e come aumentarle.

# Monitorare e ottimizzare i costi dei cluster Amazon EKS

Il monitoraggio dei costi è un aspetto essenziale della gestione dei cluster Kubernetes su Amazon EKS. Acquisendo visibilità sui costi dei cluster, è possibile ottimizzare l'utilizzo delle risorse, fissare budget e prendere decisioni relative alle implementazioni basate sui dati. Amazon EKS offre due soluzioni di monitoraggio dei costi, ognuna con vantaggi unici, per tracciare e allocare i costi in modo efficace:

Dati di ripartizione dei costi di fatturazione AWS per Amazon EKS: questa funzionalità nativa si integra perfettamente con la Console AWS per la fatturazione, consentendo di analizzare e allocare i

costi utilizzando la stessa interfaccia e gli stessi flussi di lavoro già noti utilizzati per altri servizi AWS. Con la ripartizione dei costi, è possibile ottenere approfondimenti sui costi di Kubernetes direttamente insieme alle altre spese AWS, semplificando l'ottimizzazione globale dei costi in tutto l'ambiente AWS. È anche possibile utilizzare le funzionalità di fatturazione AWS esistenti come Cost Categories e Cost Anomaly Detection per migliorare ulteriormente le capacità di gestione dei costi. Per ulteriori informazioni, consultare la pagina Understanding split cost allocation data della Guida per l'utente di Gestione dei costi e fatturazione AWS.

Kubecost: Amazon EKS supporta Kubecost, uno strumento di monitoraggio dei costi di Kubernetes. Kubecost offre un approccio ricco di funzionalità e nativo di Kubernetes al monitoraggio dei costi, con analisi granulari dei costi in base alle risorse Kubernetes, consigli per l'ottimizzazione dei costi e dashboard e report pronti all'uso. Kubecost recupera inoltre dati accurati sui prezzi grazie all'integrazione con il Report di costi e utilizzo AWS, garantendo una visione precisa dei costi di Amazon EKS. Informazioni su come installare Kubecost. Consultare la pagina Kubecost di AWS Marketplace per ulteriori informazioni su come ottenere un abbonamento gratuito a Kubecost.

## Visualizza i costi per Pod in AWS fatturazione con ripartizione dei costi

### Monitoraggio dei costi utilizzando AWS dati di allocazione dei costi frazionati per Amazon EKS

Puoi utilizzare i dati di allocazione dei costi AWS frazionati per Amazon EKS per ottenere una visibilità granulare dei costi per i tuoi cluster Amazon EKS. Ciò consente di analizzare, ottimizzare e addebitare costi e utilizzo delle applicazioni Kubernetes. I costi delle applicazioni vengono allocati a unità e team aziendali singoli in base alla CPU e alle risorse di memoria di Amazon EC2 utilizzate dall'applicazione Kubernetes. I dati di ripartizione dei costi per Amazon EKS offrono visibilità sul costo per pod e consentono di aggregare i dati relativi ai costi per pod utilizzando namespace, cluster e altre primitive Kubernetes. Di seguito sono riportati alcuni esempi di primitive Kubernetes che è possibile utilizzare per analizzare i dati di ripartizione dei costi di Amazon EKS.

- Nome cluster
- Implementazione
- Namespace
- Nodo
- Nome del carico di lavoro
- Tipo di carico di lavoro

User-defined Sono supportati anche i tag di allocazione dei costi. Per ulteriori informazioni sull'utilizzo dei dati di allocazione dei costi frazionati, vedere Informazioni sui dati di allocazione dei costi frazionati nella AWS Billing User Guide.

### Configurare i report di costi e utilizzo

Puoi attivare i dati di allocazione dei costi frazionati per EKS nella Console di gestione dei costi, nell'interfaccia a riga di AWS comando o negli SDK. AWS

Utilizzare la seguente procedura per i dati di ripartizione dei costi:

1. Attiva i Dati di ripartizione dei costi. Per ulteriori informazioni, vedere Abilitazione dei dati di allocazione dei costi frazionati nella Guida per l'utente del rapporto AWS sui costi e l'utilizzo.
2. Includere i dati in un report nuovo o esistente.
3. Visualizzare il report. Per visualizzare i file di report in Amazon Simple Storage Service, puoi utilizzare la console Gestione costi e fatturazione.

## Installare Kubecost

Amazon EKS supporta Kubecost, che è possibile utilizzare per monitorare i costi suddivisi per risorse Kubernetes compresi pod, nodi, namespace ed etichette. Questo argomento illustra l'installazione di Kubecost e l'accesso alla relativa dashboard.

Amazon EKS fornisce un pacchetto AWS ottimizzato di Kubecost per la visibilità dei costi dei cluster. Puoi utilizzare i tuoi contratti di AWS supporto esistenti per ottenere assistenza. Per ulteriori informazioni sulle versioni di Kubecost disponibili, consultare the section called "Informazioni su Kubecost".

> **Note**
>
> Kubecost v3 introduce importanti miglioramenti architetturali, tra cui prestazioni notevolmente più veloci e funzionalità di automazione avanzate. Scopri di più su Kubecost v3.
> Kubecost v2 introduce molte nuove funzionalità importanti. Scopri di più su Kubecost v2.

Per ulteriori informazioni su Kubecost, consultare la documentazione di Kubecost e le domande frequenti.

## Installare il pacchetto Kubecost ottimizzato per Amazon EKS

È possibile utilizzare una delle seguenti procedure per installare il pacchetto Kubecost ottimizzato per Amazon EKS:

- Prima di iniziare, si consiglia di consultare Kubecost - Panoramica sull'architettura per capire come funziona Kubecost su Amazon EKS.
- Se non si conosce Amazon EKS, consigliamo di utilizzare il componente aggiuntivo Amazon EKS perché semplifica l'installazione del pacchetto Kubecost ottimizzato per Amazon EKS. Per ulteriori informazioni, consultare la pagina Deploying Kubecost on an Amazon EKS cluster using Amazon EKS add-on.
- Per personalizzare l'installazione, è possibile configurare il pacchetto Kubecost ottimizzato per Amazon EKS con Helm. Per ulteriori informazioni, consultare Deploying Kubecost on an Amazon EKS cluster using Helm nella documentazione di Kubecost.

**Important**

Per Kubecost v3, la posizione del grafico Helm è cambiata in. `public.ecr.aws/kubecost/kubecost` Se stai effettuando l'aggiornamento dalla v2, aggiorna di conseguenza i riferimenti del tuo repository Helm.

**Note**

Per le implementazioni multi-cluster con Kubecost v3, è necessario lo storage a S3-compatible oggetti (S3 per i clienti EKS) per l'archiviazione delle metriche.AWS Questo sostituisce lo storage utilizzato nella versione v2. Prometheus-compatible Per ulteriori informazioni, consulta Multi-Cluster Installazione nella documentazione di Kubecost.

## Accedere alla dashboard Kubecost

Una volta completata la configurazione del pacchetto Kubecost ottimizzato per Amazon EKS, puoi accedere alla dashboard Kubecost. Per ulteriori informazioni, consulta the section called "Accedere alla dashboard Kubecost".

# Accedere alla dashboard Kubecost

## Prerequisiti

1. Assicurati che gli stati Kubecost-related dei Pod siano «In esecuzione».

```
kubectl get pods --namespace kubecost
```

## Accedere alla dashboard Kubecost

1. Sul dispositivo, abilitare l'inoltro della porta per esporre la dashboard Kubecost.
   - Se Kubecost v3 è installato utilizzando Helm:

     ```
     kubectl port-forward deployment/kubecost-frontend 9090 --namespace kubecost
     ```

   - Se Kubecost v1 o v2 è installato utilizzando Helm:

     ```
     kubectl port-forward deployment/kubecost-cost-analyzer 9090 --namespace kubecost
     ```

   - Se Kubecost è installato utilizzando il componente aggiuntivo Amazon EKS:

     ```
     kubectl port-forward deployment/cost-analyzer 9090 --namespace kubecost
     ```

     In alternativa, è possibile utilizzare AWS Load Balancer Controller per esporre Kubecost e utilizzare Amazon Cognito per l'autenticazione, l'autorizzazione e la gestione degli utenti. Per ulteriori informazioni, consultare l'articolo How to use Application Load Balancer and Amazon Cognito to authenticate users for your Kubernetes web apps.
2. Sullo stesso dispositivo su cui hai completato il passaggio precedente, apri un browser Web e inserisci il seguente indirizzo.

   ```
   http://localhost:9090
   ```

   Viene visualizzata la pagina Panoramica di Kubecost nel browser. Kubecost potrebbe impiegare 5-10 minuti (o più) per raccogliere le metriche, a seconda delle dimensioni del cluster. È possibile visualizzare la spesa Amazon EKS, inclusi i costi cumulativi dei cluster, costi delle risorse Kubernetes associate e spesa mensile aggregata.

3. Per tenere traccia dei costi a livello di cluster, tagga le tue risorse Amazon EKS per la fatturazione. Per ulteriori informazioni, consulta the section called "Tagging delle risorse per la fatturazione".
   - Ripartizione dei costi: visualizza i costi mensili di Amazon EKS e i costi cumulativi per ciascuno dei tuoi namespace e altre dimensioni negli ultimi sette giorni. Questo è utile per capire quali parti della tua applicazione contribuiscono alla spesa di Amazon EKS.
   - Risorse: visualizza i costi degli asset dell' AWS infrastruttura associati alle tue risorse Amazon EKS.

## Informazioni su Kubecost

Amazon EKS fornisce un pacchetto AWS ottimizzato di Kubecost per la visibilità dei costi dei cluster. Amazon EKS supporta Kubecost, che è possibile utilizzare per monitorare i costi suddivisi per risorse Kubernetes compresi pod, nodi, namespace ed etichette.

Questo argomento illustra le versioni di Kubecost disponibili e le differenze tra i vari livelli. EKS supporta Kubecost versione 1, versione 2 e versione 3. Ogni versione è disponibile in diversi livelli. È possibile utilizzare il pacchetto Kubecost ottimizzato per Amazon EKS per i cluster Amazon EKS senza costi aggiuntivi. È possibile che ti venga addebitato un costo per l'utilizzo di AWS servizi associati, come Amazon Managed Service for Prometheus. Inoltre, puoi utilizzare i contratti di AWS assistenza esistenti per ottenere assistenza.

In qualità di amministratore della piattaforma Kubernetes e leader finanziario, è possibile usare Kubecost per visualizzare una ripartizione degli addebiti di Amazon EKS, allocare i costi e ottenere il chargeback di unità organizzative come i team responsabili delle applicazioni. È possibile fornire ai team interni e alle unità aziendali dati sui costi trasparenti e accurati basati sulla AWS fattura effettiva. Inoltre, puoi anche ottenere consigli personalizzati per l'ottimizzazione dei costi in base all'ambiente dell'infrastruttura e ai modelli di utilizzo all'interno dei cluster. Per ulteriori informazioni su Kubecost, consultare la documentazione di Kubecost.

Qual è la differenza tra il pacchetto personalizzato di Kubecost e la versione gratuita di Kubecost (nota anche come)? OpenCost

AWS e Kubecost hanno collaborato per offrire una versione personalizzata di Kubecost. Questa versione include un sottoinsieme di funzionalità commerciali senza costi aggiuntivi. Consulta le tabelle seguenti per le funzionalità incluse nel pacchetto personalizzato di Kubecost.

### Kubecost v3

Qual è la differenza tra Kubecost v2 e v3?

Kubecost 3.0 è un importante aggiornamento dell'architettura che offre prestazioni notevolmente più veloci, scalabilità migliorata e funzionalità di ottimizzazione proattiva. La modifica più significativa è la migrazione a un ClickHouse database, che sostituisce DuckDB dalla versione 2.8, che fornisce query sostanzialmente più veloci e prestazioni più affidabili su larga scala. Kubecost 3.0 introduce anche un agente unificato che combina le funzionalità Kubecost e Cloudability, eliminando la dipendenza da Prometheus e riducendo l'ingombro di memoria mantenendo la compatibilità. OpenCost

> **Important**
>
> Consulta la documentazione di Kubecost prima di eseguire l'aggiornamento alla v3. La migrazione dalla v2 richiede un'attenta pianificazione e può influire sulla disponibilità dei report durante la transizione. La posizione della mappa di Helm è stata modificata in. `public.ecr.aws/kubecost/kubecost`

Principali miglioramenti architettonici nella v3:

- ClickHouse Database: sostituisce DuckDB per query notevolmente più veloci e una migliore scalabilità
- Unified Agent: combina le funzionalità di Kubecost e Cloudability, eliminando la dipendenza da Prometheus
- S3-Compatible Storage per Multi-Cluster: per le implementazioni multi-cluster, la versione 3 utilizza lo storage di S3-compatible oggetti (AWS S3 per i clienti EKS) anziché lo storage Prometheus-compatible come Amazon Managed Service for Prometheus. L' FinOps agente estrae i parametri dall'API Kubernetes e li trasferisce allo S3-compatible storage, quindi Aggregator estrae i dati, esegue le fasi di derivazione e visualizza i risultati nel frontend. Per ulteriori informazioni, Multi-Cluster consulta la Guida all'installazione e ai cluster secondari nella documentazione di Kubecost.
- Ingombro di memoria ridotto: requisiti di risorse notevolmente inferiori pur mantenendo la funzionalità
- Architettura semplificata: topologia Single-container pod per scalabilità indipendente e maggiore resilienza
- Automazione avanzata: dimensionamento automatizzato delle richieste dei container con consapevolezza di più cluster e profili personalizzati

Vantaggi del pacchetto ottimizzato per Amazon EKS nella versione 3:

Il pacchetto Kubecost ottimizzato per Amazon EKS continua a essere disponibile senza costi aggiuntivi ed è esente dal nuovo limite di spesa di 100.000 USD introdotto nel piano gratuito Kubecost v3. Gli utenti EKS mantengono l'accesso completo a tutte le funzionalità di spesa di Kubernetes indipendentemente dai livelli di spesa.

Confronto delle funzionalità principali:

| Funzionalità | Kubecost gratuito, livello 3.0 | Pacchetto Kubecost 3.0 ottimizzato per Amazon EKS | Kubecost Enterprise 3.0 |
|---|---|---|---|
| Visibilità costo del cluster | Cluster illimitati, con un costo limitato a 100.000 USD, da spendere in 30 giorni | Multicluster unificato senza limiti di spesa | Cluster unificati e illimitati in un numero illimitato di ambienti (ad esempio multi-cloud) |
| Backend del database | ClickHouse (locale) | ClickHouse con S3-compatible storage per metriche multi-cluster | ClickHouse con opzioni di database personalizzate |
| Performance | Query notevolmente più veloci rispetto alla v2 | Query notevolmente più veloci rispetto alla v2 | Query notevolmente più veloci rispetto alla v2 |
| Impronta di memoria | Ridotto rispetto alla v2 (nessuna dipendenza da Prometheus) | Ridotto rispetto alla v2 (nessuna dipendenza da Prometheus) | Ridotto rispetto alla v2 (nessuna dipendenza da Prometheus) |
| Dimensionamento automatico delle richieste di container | Disponibile (limitato a 250 core) | Disponibile senza limiti di base | Disponibile senza limiti di base |
| Limiti di spesa | $100.000 USD in 30 giorni | Nessun limite di spesa | Nessun limite di spesa |

| Funzionalità | Kubecost gratuito, livello 3.0 | Pacchetto Kubecost 3.0 ottimizzato per Amazon EKS | Kubecost Enterprise 3.0 |
|---|---|---|---|
| Automazione di Multi-cluster | Limitato | Conoscenza completa di più cluster con messaggistica sicura | Conoscenza completa di più cluster con messaggistica sicura |

## Kubecost v2

Qual è la differenza tra Kubecost v1 e v2?

Kubecost 2.0 è un grande aggiornamento rispetto alle versioni precedenti e include nuove importanti funzionalità tra cui un nuovo backend dell'API. Tenere presente che le API Allocation e Assets sono completamente compatibili con le versioni precedenti. Consultare la documentazione di Kubecost per garantire una transizione fluida. Per l'elenco completo dei miglioramenti, consultare l'annuncio di Kubecost v2.0 e le note di rilascio complete.

> **Important**
>
> Consultare la documentazione di Kubecost prima dell'aggiornamento. L'aggiornamento può compromettere la disponibilità dei report.

Confronto delle funzionalità principali:

| Funzionalità | Kubecost 2.0 livello gratuito | Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS | Kubecost 2.0 Enterprise |
|---|---|---|---|
| Visibilità costo del cluster | Cluster illimitati fino a 250 core | Multi-cluster unificato senza limiti core se integrato con il servizio gestito da Amazon per Prometheus | Cluster unificati e illimitati in un numero illimitato di ambienti (ad esempio multi-cloud) |

| Funzionalità | Kubecost 2.0 livello gratuito | Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS | Kubecost 2.0 Enterprise |
|---|---|---|---|
| Implementazione | Utente ospitato | Utente ospitato | Ospitata dall'utente, ospitata da Kubecost (tenant dedicato), SaaS |
| Database supportati | Prometheus locale | Servizio gestito da Amazon per Prometheus o Prometheus locale | Qualsiasi variante e database personali zzato Prometheus |
| Supporto per la conservazione del database (parametri non elaborati) | 15 giorni | Dati storici illimitati | Dati storici illimitati |
| API Kubecost e conservazione UI (ETL) | 15 giorni | 15 giorni | Illimitato |
| Visibilità del cloud ibrido | - | Cluster Amazon EKS e Amazon EKS Anywhere | Multi-cloud e cloud ibrido |
| Avvisi e report ricorrenti | Supporto solo sul cluster primario, limitato a 250 core | Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altri avvisi per tutti i cluster | Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altri avvisi per tutti i cluster |
| Report salvati. | - | Report che utilizzano 15 giorni di parametri | Report che utilizzano dati storici e parametri illimitati |

| Funzionalità | Kubecost 2.0 livello gratuito | Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS | Kubecost 2.0 Enterprise |
|---|---|---|---|
| Integrazione della fatturazione cloud | Supporto solo sul cluster primario, limitato a 250 core | Supporto tariffario personalizzato per AWS (inclusi più cluster e più account) | Supporto tariffario personalizzato per qualsiasi cloud |
| Suggerimenti per i risparmi | Supporto solo sul cluster primario, limitato a 250 core | Approfondimenti sul cluster primario, ma senza limite di 250 core | Multi-cluster approfondimenti |
| Governance: audit | - | - | Verifica gli eventi relativi ai costi storici |
| Supporto Single Sign-On (SSO) | - | Amazon Cognito supportato | Okta, Auth0, PingID e qualsiasi altra cosa KeyCloak personali zzata |
| Role-based controllo degli accessi (RBAC) con SAML 2.0 | - | - | Okta, Auth0, PingID e qualsiasi altra cosa personalizzata KeyCloak |
| Formazione e onboarding aziendali | - | - | Full-service formazion e FinOps e onboardin g |
| Team | - | - | Sì |

Nuove caratteristiche:

Le seguenti funzionalità hanno limiti legati ai parametri:

- Aggregatore Kubecost

- Monitoraggio della rete
- Operazioni Kubecost
- Raccolte
- Rilevamento anomalie
- Richiesta di container Right-Sizing
- Previsioni Kubecost
- Completamento automatico per filtraggio e aggregazione

Limiti dei parametri:

| Metrica | Kubecost 2.0 livello gratuito | Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS | Kubecost 2.0 Enterprise |
|---|---|---|---|
| Dimensione del cluster | Cluster illimitati fino a 250 core | Illimitato | Illimitato |
| Conservazione dei parametri | 15 giorni | 15 giorni | Illimitato |
| Multi-cluster supporto | Non disponibile | Disponibilità | Disponibilità |
| Limiti di core | 250 core per cluster | Nessun limite di core | Nessun limite di core |

## Kubecost v1

| Funzionalità | Kubecost livello gratuito | Pacchetto Kubecost ottimizzato per Amazon EKS | Kubecost Enterprise |
|---|---|---|---|
| Distribuzione | Utente ospitato | Utente ospitato | Ospitata dall'utente o ospitata da Kubecost (SaaS) |

| Funzionalità | Kubecost livello gratuito | Pacchetto Kubecost ottimizzato per Amazon EKS | Kubecost Enterprise |
|---|---|---|---|
| Numero di cluster supportati | Illimitato | Illimitato | Illimitato |
| Database supportati | Prometheus locale | Prometheus locale o servizio gestito da Amazon per Prometheus | Prometheus, servizio gestito da Amazon per Prometheus, Cortex o Thanos |
| Supporto per la conservazione dei database | 15 giorni | Dati storici illimitati | Dati storici illimitati |
| Conservazione API Kubecost (ETL) | 15 giorni | 15 giorni | Dati storici illimitati |
| Visibilità costo del cluster | Cluster singoli | Multicluster unificato | Multicluster unificato |
| Visibilità del cloud ibrido | - | Cluster Amazon EKS e Amazon EKS Anywhere | Multi-cloud e supporto per il cloud ibrido |
| Avvisi e report ricorrenti | - | Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altro ancora | Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altro ancora |
| Report salvati. | - | Report che utilizzano dati di 15 giorni | Report che utilizzano dati storici illimitati |
| Integrazione della fatturazione cloud | Obbligatorio per ogni singolo cluster | Supporto tariffario personalizzato per AWS (inclusi più cluster e più account) | Supporto tariffario personalizzato per AWS (inclusi più cluster e più account) |

| Funzionalità | Kubecost livello gratuito | Pacchetto Kubecost ottimizzato per Amazon EKS | Kubecost Enterprise |
|---|---|---|---|
| Suggerimenti per i risparmi | Informazioni su un singolo cluster | Informazioni su un singolo cluster | Multi-cluster approfondimenti |
| Governance: audit | - | - | Verifica gli eventi relativi ai costi storici |
| Supporto Single Sign-On (SSO) | - | Amazon Cognito supportato | Okta, Auth0, PingID, KeyCloak |
| Role-based controllo degli accessi (RBAC) con SAML **2.0** | - | - | Okta, Auth0, PingID, Keycloak |
| Formazione e onboarding aziendali | - | - | Full-service formazione e onboarding FinOps |

## Domande frequenti

Consultare le seguenti domande frequenti e relative risposte sull'utilizzo di Kubecost con Amazon EKS.

Cos'è la funzionalità di conservazione dell'API Kubecost (ETL)?

La funzionalità ETL Kubecost aggrega e organizza i parametri per evidenziare la visibilità dei costi a vari livelli di granularità (come `namespace-level`, `pod-level` e `deployment-level`). Per il pacchetto Kubecost ottimizzato per Amazon EKS, i clienti ottengono dati e approfondimenti dai parametri degli ultimi 15 giorni.

Cos'è la funzionalità di avvisi e report ricorrenti? Quali avvisi e report include?

Gli avvisi Kubecost consentono ai team di ricevere aggiornamenti in tempo reale sulla spesa Kubernetes, nonché sulla spesa per il cloud. I report ricorrenti consentono ai team di ricevere visualizzazioni personalizzate della cronologia della spesa Kubernetes e per il cloud. Entrambi sono configurabili utilizzando l'UI Kubecost o i valori Helm. Supportano e-mail, Slack e Microsoft Teams.

Cosa includono i report salvati?

I report salvati Kubecost sono visualizzazioni predefinite dei parametri di costi ed efficienza. Includono il costo per cluster, namespace, etichetta e altro ancora.

Cos'è l'integrazione della fatturazione nel cloud?

L'integrazione con le API di AWS fatturazione consente a Kubecost di visualizzare i costi esterni al cluster (come Amazon S3). Inoltre, consente a Kubecost di riconciliare le previsioni interne al cluster di Kubecost con i dati di fatturazione effettivi, per tenere conto dell'utilizzo spot, dei Savings Plans e degli sconti aziendali.

Cosa includono i consigli di risparmio?

Kubecost offre approfondimenti e automazione per aiutare gli utenti a ottimizzare l'infrastruttura Kubernetes e le relative spese.

Questa funzionalità è a pagamento?

No. È possibile utilizzare il pacchetto Kubecost ottimizzato per Amazon EKS senza costi aggiuntivi. Se desideri funzionalità Kubecost aggiuntive non incluse, puoi acquistare una licenza Enterprise di Kubecost tramite AWS Marketplace o direttamente da Kubecost.

È disponibile supporto per il pacchetto Kubecost ottimizzato per Amazon EKS?

Sì, solo se si utilizza il pacchetto Kubecost ottimizzato per Amazon EKS.

Come si può richiedere supporto per il pacchetto Kubecost ottimizzato per Amazon EKS?

Puoi aprire una richiesta di supporto con il team di AWS supporto all'indirizzo Contatti AWS.

È necessaria una licenza per l'utilizzo delle funzionalità di Kubecost fornite dall'integrazione Amazon EKS?

No.

Posso integrare AWS Kubecost con Cost and Usage Report per un reporting più accurato?

Sì. Puoi configurare Kubecost in modo che acquisisca i dati dal rapporto sui AWS costi e sull'utilizzo per ottenere una visibilità accurata dei costi, inclusi sconti, prezzi Spot, prezzi delle istanze riservate e altro. Per ulteriori informazioni, consultare la pagina AWS Cloud Billing Integration della documentazione di Kubecost.

Questa versione supporta la gestione dei costi dei cluster Kubernetes autogestiti su Amazon EC2?

No. Il pacchetto Kubecost ottimizzato per Amazon EKS è compatibile solo con i cluster Amazon EKS.

Kubecost può tenere traccia dei costi di Amazon EKS su AWS Fargate?

Kubecost fa tutto il possibile per mostrare la visibilità dei costi dei cluster di Amazon EKS su Fargate, ma con una precisione inferiore rispetto ad Amazon EKS su Amazon EC2. Ciò è dovuto principalmente alla differenza nella modalità di fatturazione dell'utilizzo. Con Amazon EKS su Fargate, vengono fatturate le risorse utilizzate. Con Amazon EKS sui nodi Amazon EC2, vengono fatturate le risorse assegnate. Kubecost calcola il costo di un nodo Amazon EC2 in base alle relative specifiche, che includono CPU, RAM e archiviazione temporanea. Con Fargate, i costi vengono calcolati in base alle risorse richieste per i pod Fargate.

Come si possono ottenere aggiornamenti e nuove versioni di Kubecost?

È possibile aggiornare la versione di Kubecost utilizzando le procedure di aggiornamento standard di Helm. Per Kubecost v3, le versioni più recenti sono disponibili nella nuova posizione cartografica di Helm. `public.ecr.aws/kubecost/kubecost` Le versioni precedenti (v1 e v2) rimangono disponibili nella Amazon ECR Public Gallery.

> **Important**
>
> Quando esegui l'aggiornamento a Kubecost v3, tieni presente che la posizione della mappa di Helm è cambiata da a. `public.ecr.aws/kubecost/cost-analyzer` `public.ecr.aws/kubecost/kubecost` Aggiorna di conseguenza i riferimenti del tuo repository Helm.

La CLI di **`kubectl-cost`** è supportata? Come posso installarlo?

Sì. `Kubectl-cost` è uno strumento open source di Kubecost (licenza Apache 2.0) che offre l'accesso tramite CLI ai parametri di ripartizione dei costi di Kubernetes. Per installarlo`kubectl-cost`, vedi Installazione su GitHub.

L'interfaccia utente di Kubecost è supportata? Come posso accedervi?

Kubecost offre un pannello di controllo web a cui è possibile accedere tramite l'inoltro alla porta `kubectl`, un ingresso o un bilanciatore del carico. Puoi anche usare AWS Load Balancer Controller

per esporre Kubecost e usare Amazon Cognito per l'autenticazione, l'autorizzazione e la gestione degli utenti. Per ulteriori informazioni, consulta Come usare Application Load Balancer e Amazon Cognito per autenticare gli utenti per le tue app web Kubernetes sul blog. AWS

Il nuovo limite di spesa di 100.000 dollari in Kubecost v3 influisce sugli utenti di Amazon EKS?

No. Il limite di spesa di 100.000 USD per 30 giorni introdotto nel piano gratuito Kubecost v3 non si applica agli utenti del pacchetto Kubecost ottimizzato per Amazon EKS. Gli utenti EKS mantengono l'accesso completo a tutte le funzionalità di spesa di Kubernetes indipendentemente dai livelli di spesa.

Quali sono i principali miglioramenti delle prestazioni di Kubecost v3?

Kubecost v3 introduce sostanziali miglioramenti delle prestazioni attraverso il backend del ClickHouse database, che fornisce query notevolmente più veloci rispetto al DuckDB utilizzato nella versione 2.8. Inoltre, l'architettura unificata degli agenti elimina la dipendenza da Prometheus, riducendo l'ingombro della memoria e mantenendo al contempo la piena funzionalità e compatibilità. OpenCost

Quale backend di storage utilizza Kubecost v3 per le implementazioni multi-cluster?

Kubecost v3 utilizza lo storage a S3-compatible oggetti (AWS S3 per i clienti EKS) per lo storage di metriche multi-cluster, sostituendo lo storage utilizzato nella v2. Prometheus-compatible L' FinOps agente raccoglie le metriche dall'API Kubernetes e le invia allo storage. S3-compatible L'Aggregator recupera quindi questi dati, esegue calcoli dei costi e visualizza i risultati nel frontend. Per istruzioni dettagliate sulla configurazione multicluster, consulta la Guida Multi-Cluster all'installazione e ai cluster secondari nella documentazione di Kubecost.

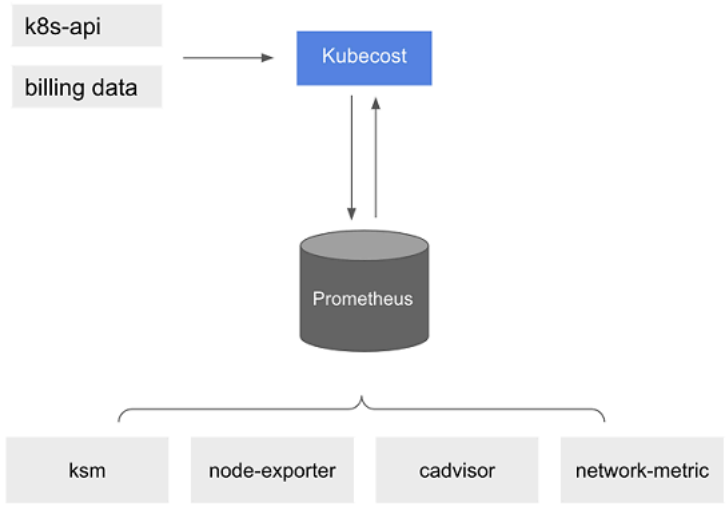
Posso effettuare l'aggiornamento direttamente da Kubecost v1 a v3?

No. L'aggiornamento diretto dalla v1 alla v3 non è supportato. È necessario prima eseguire l'aggiornamento alla v2, quindi migrare alla v3. Consulta la documentazione di Kubecost per indicazioni dettagliate sulla migrazione, poiché il processo richiede un'attenta pianificazione e può influire sulla disponibilità dei report durante la transizione.

## Funzionalità Kubecost aggiuntive

- Le seguenti funzionalità sono disponibili in Kubecost v1, v2 e v3.
  - Metriche dei costi di esportazione: il monitoraggio ottimizzato dei costi di Amazon EKS viene implementato con Kubecost. Nelle versioni 1 e 2, Kubecost si integra con Prometheus per l'archiviazione e l'elaborazione delle metriche. Nella v3, Kubecost utilizza un database per

migliorare notevolmente le prestazioni mantenendo al contempo la compatibilità. ClickHouse OpenCost Per le implementazioni multi-cluster nella v3, le metriche vengono archiviate nello storage di S3-compatible oggetti (AWS S3 per i clienti EKS) anziché nello storage. Prometheus-compatible Kubecost legge le metriche, esegue calcoli di allocazione dei costi e fornisce dati tramite le sue API e l'interfaccia utente. L'architettura varia in base alla versione ma mantiene funzionalità coerenti.




Puoi scrivere query per inserire i dati di Kubecost nel tuo attuale sistema di business intelligence per ulteriori analisi. È anche possibile usarlo come origine dati per la dashboard Grafana attuale al fine di visualizzare i costi dei cluster Amazon EKS che i team interni conoscono. Per saperne di più su come scrivere query, consulta la documentazione di OpenCost configurazione o usa come riferimenti i modelli JSON Grafana di esempio nel repository GitHub Kubecost.

- AWS Integrazione dei report su costi e utilizzo: per eseguire calcoli di allocazione dei costi per il cluster Amazon EKS, Kubecost recupera le informazioni pubbliche sui prezzi di AWS servizi e AWS risorse dall'API Price List. AWS Puoi anche integrare AWS Kubecost con Cost and Usage Report per migliorare l'accuratezza delle informazioni sui prezzi specifiche del tuo account. AWS Queste informazioni includono programmi di discount aziendali, utilizzo di istanze riservate, Savings Plans e utilizzo spot. Per saperne di più su come funziona l'integrazione del AWS Cost and Usage Report, consulta AWS Cloud Billing Integration nella documentazione di Kubecost.

## Visualizza l'utilizzo delle risorse con Kubernetes Metrics Server

Kubernetes Metrics Server è un aggregatore dei dati di utilizzo delle risorse nel cluster e non viene implementato nei cluster Amazon EKS per impostazione predefinita. Per ulteriori informazioni,

consulta Kubernetes Metrics Server on. GitHub Il Metrics Server viene comunemente utilizzato da altri componenti aggiuntivi di Kubernetes, come le implementazioni di pod Scale con Horizontal Pod Autoscaler o Kubernetes Dashboard. Per ulteriori informazioni, consulta la pipeline di parametri delle risorse nella documentazione di Kubernetes. In questo argomento viene descritto come implementare Kubernetes Metrics Server sul cluster Amazon EKS.

> **Important**
>
> I parametri sono progettati per l'analisi puntuale e non costituiscono una fonte accurata per l'analisi storica. Non possono essere utilizzati come soluzione di monitoraggio o per altri scopi di dimensionamento non automatico. Per ulteriori informazioni sul monitoraggio, consulta la pagina Monitoraggio dei cluster.

## Considerazioni

- Se distribuisci manualmente Kubernetes Metrics Server sui nodi Fargate utilizzando il manifesto, configura l'implementazione `metrics-server` per utilizzare una porta diversa da quella predefinita di `10250`. Questa porta è riservata a Fargate. La versione del componente aggiuntivo di Amazon EKS di Metrics Server è preconfigurata per utilizzare la porta `10251`.
- Assicurati che i gruppi di sicurezza e gli ACL di rete consentano la porta `10250` tra i pod `metrics-server` e tutti gli altri nodi e pod. Il Kubernetes Metrics Server utilizza ancora la porta `10250` per raccogliere metriche da altri endpoint del cluster. Se esegui l'implementazione sui nodi Fargate, consenti sia la porta Metrics Server alternativa configurata che la porta `10250`.

## Implementa come componente aggiuntivo della community con Amazon EKS Add-ons

Novità: ora puoi distribuire Metrics Server come componente aggiuntivo della community utilizzando la console AWS o le API di Amazon EKS.

### Implementa con AWS console

1. Apri il tuo cluster EKS nella console AWS
2. Dalla scheda "Add-ons", seleziona Ottieni di più Add-ons.
3. Dalla sezione "Community add-ons", seleziona Metrics Server, quindi Avanti

4. EKS determina la versione appropriata del componente aggiuntivo per il cluster. È possibile modificare la versione utilizzando il menu a discesa Versione.
5. Seleziona Avanti e poi Crea per installare il componente aggiuntivo.

### Risorse aggiuntive

Ulteriori informazioni su the section called "Componenti aggiuntivi della community".

Puoi installare o aggiornare i componenti aggiuntivi della community allo stesso modo degli altri Amazon EKS Add-ons.

- the section called "Creare un componente aggiuntivo"
- the section called "Aggiornamento di un componente aggiuntivo"
- the section called "Rimozione di un componente aggiuntivo"

## Implementa con manifesto

Novità: ora puoi distribuire Metrics Server come componente aggiuntivo della community utilizzando la console AWS o le API di Amazon EKS. Queste istruzioni di installazione del manifesto verranno archiviate.

1. implementare Metrics Server con il comando seguente:

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
   ```

   Se utilizzi Fargate, dovrai modificare questo file. Nella configurazione predefinita, Metric Server utilizza la porta 10250. Questa porta è riservata a Fargate. Sostituisci i riferimenti alla porta 10250 in components.yaml con un'altra porta, ad esempio 10251.
2. Verifica che l'implementazione `metrics-server` esegua il numero di pod desiderato con il comando seguente:

   ```
   kubectl get deployment metrics-server -n kube-system
   ```

   Di seguito viene riportato un output di esempio.

   ```
   NAME            READY   UP-TO-DATE   AVAILABLE   AGE
   ```

```
metrics-server   1/1     1            1           6m
```

3. Verifica che il server delle metriche funzioni visualizzando l'utilizzo delle risorse () CPU/memory dei nodi.

```
kubectl top nodes
```

4. Se ricevi il messaggio di errore `Error from server (Forbidden)`, devi aggiornare la configurazione RBAC di Kubernetes. La tua identità Kubernetes RBAC richiede autorizzazioni sufficienti per leggere le metriche del cluster. Verifica le autorizzazioni minime richieste per l'API Kubernetes per la lettura delle metriche. GitHub Scopri come concedere alle identità AWS IAM, come i ruoli, l'accesso alle API Kubernetes.

# Implementazione di applicazioni con Helm su Amazon EKS

Il programma di gestione del pacchetto Helm per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes. Per ulteriori informazioni, consultare la documentazione di Helm. Questo argomento consente di installare ed eseguire i file binari Helm per consentire l'installazione e la gestione di grafici utilizzando la CLI Helm nel sistema locale.

> **Important**
>
> Prima di installare grafici Helm nel cluster Amazon EKS, è necessario configurare `kubectl` per l'utilizzo di Amazon EKS. Se non hai già eseguito questa operazione, consultare the section called "Accedere al cluster con kubectl" prima di continuare. Se il comando seguente va a buon fine per il cluster, la configurazione è corretta.
>
> ```
> kubectl get svc
> ```

1. Eseguire il comando appropriato per il sistema operativo client.
   - Se utilizzi macOS con Homebrew, installa i file binari con il comando seguente.

     ```
     brew install helm
     ```

   - Per ulteriori opzioni di installazione, consulta Installing Helm nella documentazione di Helm.

> **Note**
>
> Se ricevi un messaggio che indica la necessità di installare prima `openssl`, è possibile installarlo con il seguente comando.

```
sudo yum install openssl
```

1. Per prelevare il nuovo binario nel tuoPATH, chiudi la finestra corrente del terminale e apri una nuova.
2. Guarda la versione di Helm che hai installato.

   ```
   helm version --template='{{ .Version }}{{ "\n" }}'
   ```

   Di seguito viene riportato un output di esempio.

   ```
   v3.17.2
   ```

3. Assicurati che la versione installata sia compatibile con la tua versione del cluster. Per ulteriori informazioni, consulta la versione supportata di Skew. Ad esempio, se stai utilizzando`3.17.x`, la versione di Kubernetes supportata dovrebbe essere compresa nell'intervallo da a. `1.29.x` `1.32.x`
4. A questo punto, è possibile eseguire qualsiasi comando Helm (ad esempio `helm install` *`chart-name`* ) per installare, modificare, eliminare o interrogare i grafici Helm nel cluster. Se sei un nuovo utente di Helm e non hai un grafico specifico da installare, puoi effettuare le seguenti operazioni:
   - Sperimentare installando un grafico di esempio. Consulta Install an Example Chart nella Quickstart Guide di Helm.
   - Crea un grafico di esempio e invialo ad Amazon ECR. Per ulteriori informazioni, consultare Invio di un grafico Helm nella Guida per l'utente del registro del container di Amazon Elastic.
   - Installa un grafico Amazon EKS dal repository https://github.com/aws/eks-charts#eks-charts GitHub eks-charts o da. ArtifactHub

# Organizzazione delle risorse Amazon EKS con tag

È possibile utilizzare tag per aiutarti a gestire le risorse Amazon EKS. In questo argomento viene fornita una panoramica della funzione tag e viene illustrato come creare tag.

Argomenti


- Nozioni di base sui tag
- Tagging delle risorse
- Limitazioni applicate ai tag
- Tagging delle risorse per la fatturazione
- Utilizzo di tag tramite la console
- Lavorare con i tag utilizzando la CLI, l'API o eksctl


> **Note**
>
> I tag sono un tipo di metadati separati dalle etichette e dalle annotazioni di Kubernetes. Per ulteriori informazioni sugli altri tipi di metadati, consulta le sezioni seguenti nella documentazione di Kubernetes:
>
> - Etichette e selettori
> - Annotazioni

## Nozioni di base sui tag

Un tag è un'etichetta che si assegna a una AWS risorsa. Ciascun tag è formato da una chiave e da un valore opzionale.

Con i tag, puoi classificare le tue AWS risorse. Ad esempio, puoi categorizzarle in base a scopo, proprietario o ambiente. In presenza di un numero elevato di risorse dello stesso tipo, si possono usare i tag assegnati a una risorsa specifica per identificarla rapidamente. Ad esempio, puoi definire un set di tag per i cluster Amazon EKS per monitorare il proprietario di ogni cluster e il livello di stack. Consigliamo di definire un set coerente di chiavi di tag per ciascun tipo di risorsa. È possibile cercare e filtrare le risorse in base ai tag aggiunti.

Dopo aver aggiunto un tag, puoi modificarne le chiavi e i valori oppure rimuovere i tag da una risorsa in qualsiasi momento. Se elimini una risorsa, verranno eliminati anche tutti i tag a essa associati.

I tag non hanno alcun significato semantico per Amazon EKS e vengono interpretati rigorosamente come una stringa di caratteri. È possibile impostare il valore di un tag su una stringa vuota. Tuttavia, non è possibile impostare il valore di un tag su null. Se aggiungi un tag con la stessa chiave di un tag esistente a una risorsa specifica, il nuovo valore sovrascrive quello precedente.

Se utilizzi AWS Identity and Access Management (IAM), puoi controllare quali utenti del tuo AWS account sono autorizzati a gestire i tag.

## Tagging delle risorse

I tag sono disponibili per le risorse Amazon EKS seguenti:

- cluster
- gruppi di nodi gestiti
- profili Fargate

Puoi aggiungere un tag a queste risorse utilizzando le opzioni seguenti:

- Se utilizzi la console Amazon EKS, puoi applicare tag alle risorse nuove o esistenti in qualsiasi momento. A tale scopo, è possibile utilizzare la sheda Tag nella pagina della risorsa interessata. Per ulteriori informazioni, consulta the section called "Utilizzo di tag tramite la console".
- Se utilizzi `eksctl`, puoi applicare i tag alle risorse quando vengono create utilizzando l'opzione `--tags`.
- Se utilizzi la AWS CLI, l'API Amazon EKS o un AWS SDK, puoi applicare tag a nuove risorse utilizzando il `tags` parametro sull'azione API pertinente. Puoi applicare tag a risorse esistenti utilizzando l'operazione API `TagResource`. Per ulteriori informazioni, consulta TagResource.

Quando utilizzi alcune azioni per la creazione di risorse, puoi anche specificare i tag per la risorsa nello stesso momento in cui la crei. Se i tag non possono essere applicati durante la creazione di una risorsa, la risorsa non viene creata. Mediante questo meccanismo ti assicuri che le risorse a cui desideri applicare tag al momento della creazione vengono create con tag specifici o non vengono create affatto. Se aggiungi tag alle risorse al momento della creazione, non è necessario eseguire script di assegnazione di tag personalizzati dopo la creazione di una risorsa.

I tag non si propagano ad altre risorse associate alla risorsa creata. Ad esempio, i tag del profilo di Fargate non vengono propagati ad altre risorse associate a tale profilo, come i pod pianificati con esso.

## Limitazioni applicate ai tag

Ai tag si applicano le limitazioni seguenti:

- È possibile associare un massimo di 50 tag a una risorsa.
- Le chiavi dei tag non possono essere ripetute per una risorsa. Ogni chiave di tag deve essere univoca e può avere un solo valore.
- Le chiavi possono contenere fino a 128 caratteri. UTF-8
- I valori possono contenere fino a 256 caratteri UTF-8.
- Se più AWS servizi e risorse utilizzano il tuo schema di codifica, limita i tipi di caratteri che usi. Alcuni servizi potrebbero avere restrizioni sui caratteri consentiti. I caratteri generalmente consentiti sono lettere, numeri, spazi e i simboli seguenti: `+ - = . _ : / @`.
- I valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole.
- Non utilizzare `aws:`, `AWS:` o qualsiasi combinazione di maiuscole o minuscole come prefisso per chiavi o valori. Questi sono riservati solo all' AWS uso. Non puoi modificare né eliminare le chiavi o i valori di tag con tale prefisso. I tag con questo prefisso non vengono conteggiati per il limite del numero di tag per risorsa.

## Tagging delle risorse per la fatturazione

Quando applichi i tag ai cluster Amazon EKS, puoi utilizzarli per l'allocazione dei costi nel tuo Rapporti su costi e utilizzo. I dati di misurazione nel Rapporti su costi e utilizzo mostrano l'utilizzo in tutti i processi dei cluster di Amazon EKS. Per ulteriori informazioni, consulta il rapporto su AWS costi e utilizzo nella AWS Billing User Guide.

Il tag di allocazione dei costi AWS generato, in particolare`aws:eks:cluster-name`, consente di suddividere i costi delle istanze Amazon EC2 per singolo cluster Amazon EKS in Cost Explorer. Tuttavia, questo tag non rileva le spese del piano di controllo (control-plane). Il tag viene aggiunto automaticamente alle istanze Amazon EC2 che fanno parte di un cluster Amazon EKS. Questo comportamento si verifica indipendentemente dal fatto che venga eseguito il provisioning delle istanze tramite gruppi di nodi gestiti da Amazon EKS, Karpenter o direttamente con Amazon EC2. Questo tag specifico non viene conteggiato ai fini del limite di 50 tag. Per utilizzare il tag, il proprietario dell'account deve attivarlo nella console di AWS fatturazione o utilizzando l'API. Quando il proprietario dell'account di gestione di un' AWS organizzazione attiva il tag, questo viene attivato anche per tutti gli account membri dell'organizzazione.

Puoi organizzare le informazioni di fatturazione in base alle risorse con gli stessi valori di chiave di tag. Puoi ad esempio applicare tag a numerose risorse con un nome di applicazione specifico, quindi organizzare le informazioni di fatturazione. In questo modo, puoi visualizzare il costo totale dell'applicazione in più servizi. Per ulteriori informazioni sull'impostazione di un report di allocazione dei costi con tag, consulta The Monthly Cost Allocation Report nella AWS Billing User Guide.

> **Note**
>
> Se hai appena abilitato la reportistica, i dati relativi al mese corrente saranno disponibili per la visualizzazione dopo 24 ore.

Cost Explorer è uno strumento di reporting disponibile come parte del AWS piano gratuito. È possibile utilizzare Cost Explorer per visualizzare i grafici delle tue risorse Amazon EKS degli ultimi 13 mesi. Puoi anche prevedere le spese per i prossimi tre mesi. Puoi vedere i modelli di quanto spendi in AWS risorse nel tempo. Ad esempio, puoi utilizzarlo per identificare aree che richiedono ulteriore studio e visualizzare le tendenze che puoi utilizzare per comprendere i costi. Puoi anche specificare intervalli di tempo per i dati e visualizzare i dati temporali per mese o per giorno.

## Utilizzo di tag tramite la console

Con la console Amazon EKS è possibile gestire i tag associati ai gruppi di nodi gestiti o ai cluster nuovi o esistenti.

Quando selezioni una pagina relativa alle risorse nella console Amazon EKS, viene visualizzato l'elenco delle risorse corrispondenti. Ad esempio, se nel riquadro di navigazione sinistro si seleziona Cluster, nella console viene visualizzato l'elenco dei cluster Amazon EKS. Quando si seleziona una risorsa in uno di questi elenchi, (ad esempio un cluster specifico), se la risorsa supporta i tag, sarà possibile visualizzare e gestire i tag nella scheda Tag.

Puoi anche utilizzare Tag Editor in Console di gestione AWS, che fornisce un modo unificato per gestire i tag. Per ulteriori informazioni, consulta Taggare le AWS risorse con Tag Editor nella Guida per l'utente di AWS Tag Editor.

### Aggiunta di tag a una risorsa in fase di creazione

É possibile aggiungere tag ai gruppi di nodi gestiti, ai cluster Amazon EKS ed ai profili Fargate durante la creazione. Per ulteriori informazioni, consulta the section called “Creazione di un cluster ”.

### Aggiunta ed eliminazione di tag in una risorsa

Puoi aggiungere o eliminare i tag associati ai cluster direttamente dalla pagina della risorsa.

1. Aprire la Console Amazon EKS.
2. Nella barra di navigazione, seleziona la AWS regione da utilizzare.
3. Nel pannello di navigazione a sinistra, seleziona Cluster.
4. Scegliere un cluster specifico.
5. Scegliere la scheda Tags (Tag) quindi scegliere Manage tags (Gestisci tag).
6. Nella pagina Gestisci i tag, aggiungi o elimina i tag in base alle esigenze.
   - Per aggiungere un tag, scegli Add tag (Aggiungi tag). Specificare la chiave e il valore per ogni tag.
   - Per eliminare un tag, scegli Rimuovi tag.
7. Ripeti la procedura per ogni tag da aggiungere o eliminare.
8. Scegliere Update (Aggiorna) per terminare.

## Lavorare con i tag utilizzando la CLI, l'API o eksctl

Usa i seguenti comandi AWS CLI o le operazioni API di Amazon EKS per aggiungere, aggiornare, elencare ed eliminare i tag per le tue risorse. È possibile utilizzare solo `eksctl` per aggiungere tag e contemporaneamente creare le nuove risorse con un unico comando.

| Operazione | AWS CLI | AWS Strumenti per Windows PowerShell | Azione API |
|---|---|---|---|
| Aggiungere sovrascrivere uno o più tag. | tag-resource | Add-EKSResourceTag | TagResource |
| Eliminare uno o più tag. | untag-resource | Remove-EKSResourceTag | UntagResource |

Gli esempi seguenti mostrano come etichettare o rimuovere i tag dalle risorse utilizzando la AWS CLI.

Esempio 1: applicazione di un tag a un cluster esistente

Il comando seguente applica un tag a un cluster esistente.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

Esempio 2: rimozione di un tag a un cluster esistente

Il comando seguente elimina un tag da un cluster esistente.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Esempio 3: elencazione dei tag di una risorsa

Il comando seguente elenca i tag associati a una risorsa esistente.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

Quando utilizzi alcune azioni per la creazione di risorse, puoi anche specificare i tag per la risorsa nello stesso momento in cui la crei. Le seguenti operazioni supportano l'assegnazione di tag durante la creazione di una risorsa.

| Operazione | AWS CLI | AWS Strumenti per Windows PowerShell | Azione API | eksctl |
|---|---|---|---|---|
| Creazione di un cluster | create-cluster | New-EKSCluster | CreateCluster | `create cluster` |
| Creazione di un gruppo di nodi gestito* | create-nodegroup | New-EKSNodegroup | CreateNodegroup | `create nodegroup` |
| Creazione di un profilo Fargate | create-fargate-profile | New-EKSFargateProfile | CreateFargateProfile | `create fargateprofile` |

- Se desideri taggare anche le istanze Amazon EC2 durante la creazione di un gruppo di nodi gestito, crea il gruppo di nodi gestito utilizzando un modello di avvio. Per ulteriori informazioni, consulta the section called "Assegnazione di tag a istanze Amazon EC2". Se le istanze esistono già, è possibile taggare manualmente le istanze. Per ulteriori informazioni, consulta Tagging your resources nella Guida per l'utente di Amazon EC2.

# Visualizzazione e gestione di quote di servizio di Amazon EKS e Fargate

Amazon EKS si è integrato con Service Quotas, un AWS servizio che puoi utilizzare per visualizzare e gestire le quote da una posizione centrale. Per ulteriori informazioni, consulta Cos'è Service Quotas? nella Guida per l'utente di Service Quotas. Con l'integrazione di Service Quotas, puoi cercare rapidamente il valore delle tue quote di servizio Amazon EKS e AWS Fargate utilizzando la CLI e. Console di gestione AWS AWS

## Visualizza le quote dei servizi EKS nel Console di gestione AWS

1. Apri la console Service Quotas.
2. Nel riquadro di navigazione a sinistra, scegli  AWS servizi.
3. Dall'elenco dei  AWS servizi, cerca e seleziona Amazon Elastic Kubernetes Service (Amazon EKS) o Fargate. AWS

   Nell'elenco delle quote di servizio, puoi visualizzare il nome della quota di servizio, il valore applicato (se disponibile), la quota AWS predefinita e se il valore della quota è regolabile.
4. Per visualizzare ulteriori informazioni su una quota di servizio, ad esempio la descrizione, scegli il nome della quota.
5. (Facoltativo) Per richiedere un aumento della quota, seleziona la quota che desideri aumentare, seleziona Richiedi un aumento della quota, inserisci o seleziona le informazioni richieste e seleziona Richiedi.

Per utilizzare meglio le quote di servizio utilizzando la Console di gestione AWS, consulta la Service Quotas User Guide. Per richiedere un aumento delle quote, consultare Richiesta di aumento delle quote nella Guida per l'utente di Service Quotas.

## Visualizza le quote dei servizi EKS con la CLI AWS

Eseguire questo comando per visualizzare le quote Amazon EKS.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

Eseguire questo comando per visualizzare le quote di Fargate.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

> **Note**
>
> La quota restituita è il numero di attività Amazon ECS o Amazon EKS Pods che possono essere eseguite contemporaneamente su Fargate in questo account nella regione corrente. AWS

Per lavorare di più con le quote di servizio utilizzando la AWS CLI, consulta service-quotas nel AWS CLI Command Reference. Per richiedere un aumento della quota, consulta il request-service-quota-increasecomando nella  AWS CLI Command Reference.

## Service Quotas di Amazon EKS

AWS consiglia di utilizzare il Console di gestione AWS per visualizzare le quote correnti. Per ulteriori informazioni, consulta the section called “Visualizza le quote dei servizi EKS nel Console di gestione AWS”.

Per visualizzare le quote di servizio EKS predefinite, consulta gli endpoint e le quote di Amazon Elastic Kubernetes Service nella Guida generale. AWS

Queste quote del servizio sono elencate in Amazon Elastic Kubernetes Service (Amazon EKS) nella console Service Quotas. Per chiedere un incremento della quota per i valori indicati come regolabili, consulta Richiesta di incremento di una quota nella Guida per l'utente di Service Quotas.

> **Note**
>
> Le modifiche ai seguenti componenti non sono supportate in Service Quotas: * Associazioni di Pod Identity per cluster. the section called “Pod Identity”Per i limiti, consulta. * CIDRs per reti a nodi remoti o reti pod remoti per nodi ibridi. Per i limiti, vederethe section called “Nodi ibridi”.

# AWS Quote di servizio Fargate

Il servizio  AWS Fargate nella console Service Quotas elenca diverse quote di servizio. È possibile configurare gli allarmi che avvisano quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni, consulta the section called "Creazione di un CloudWatch allarme per monitorare le metriche di utilizzo delle risorse di Fargate".

AWS I nuovi account potrebbero avere quote iniziali inferiori che possono aumentare nel tempo. Fargate monitora costantemente l'utilizzo dell'account all'interno di ciascuna AWS regione, quindi aumenta automaticamente le quote in base all'utilizzo. Puoi chiedere anche un incremento della quota per i valori indicati come regolabili. Per ulteriori informazioni, consulta Richiesta di un aumento delle quote nella Guida per l'utente di Service Quotas.

AWS consiglia di utilizzare il Console di gestione AWS per visualizzare le quote correnti. Per ulteriori informazioni, consulta the section called "Visualizza le quote dei servizi EKS nel Console di gestione AWS".

Per visualizzare le quote di servizio AWS Fargate su EKS predefinite, vedere le quote di servizio Fargate nella Guida generale. AWS

> **Note**
>
> Inoltre, Fargate applica attività Amazon ECS e quote di velocità di avvio dei pod Amazon EKS. Per ulteriori informazioni, consulta Quote di limitazione (della larghezza di banda della rete AWS Fargate nella Guida per Amazon ECS.

# Sicurezza in Amazon EKS

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. Per Amazon EKS, AWS è responsabile del piano di controllo Kubernetes, che include i nodi del piano di controllo e il database. `etcd` I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon EKS, consultare Servizi AWS coperti dal programma di conformità.
- Sicurezza nel cloud - Le seguenti aree sono sotto la responsabilità dell'utente.
  - La configurazione di sicurezza del piano dei dati, compresa la configurazione dei gruppi di sicurezza che autorizzano il passaggio del traffico dal piano di controllo Amazon EKS al VPC del cliente
  - La configurazione dei nodi di lavoro e degli stessi container
  - Il sistema operativo che ospita il nodo di lavoro (inclusi gli aggiornamenti e le patch di sicurezza)
  - Altri software applicativi associati:
    - La configurazione e la gestione dei controlli di rete, come le regole del firewall
    - La gestione di identità a livello di piattaforma e la gestione degli accessi, con o in aggiunta alle IAM
  - La riservatezza dei dati, i requisiti dell'azienda e le leggi e le normative applicabili.

Amazon EKS è certificato da svariati programmi di conformità per applicazioni regolamentate e sensibili. Amazon EKS è conforme a SOC, PCI, ISO, FedRAMP-Moderate, IRAP, C5, K-ISMS, ENS High, OSPAR, HITRUST CSF, ed è un servizio idoneo allo standard HIPAA. Per ulteriori informazioni, consulta Gestione dell'accesso.

Questa documentazione aiuta a comprendere come applicare il modello di responsabilità condivisa quando si utilizza Amazon EKS. Gli argomenti seguenti descrivono come configurare Amazon EKS

per soddisfare gli obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Amazon EKS.

> **Note**
>
> I container Linux sono costituiti da gruppi di controllo (cgroup) e namespace che aiutano a limitare l'accesso a cui un container può accedere, ma tutti i contenitori condividono lo stesso kernel Linux dell'istanza Amazon host. EC2 L'esecuzione di un container come utente root (UID 0), o la concessione di un accesso al container a risorse host o namespace, come la rete host o il namespace PID host sono fortemente sconsigliati, in quanto ciò riduce l'efficacia dell'isolamento fornito dai container.

Argomenti


- Protezione dei cluster Amazon EKS con le best practice
- Analizzare le vulnerabilità in Amazon EKS
- Convalida della conformità per cluster di Amazon EKS
- Considerazioni di sicurezza per Amazon Elastic Container Service per Kubernetes
- Considerazioni sulla sicurezza per Kubernetes
- Considerazioni sulla sicurezza per la modalità automatica di Amazon EKS
- Considerazioni sulla sicurezza per EKS Capabilities
- Identity and Access Management per Amazon EKS


## Protezione dei cluster Amazon EKS con le best practice

Le best practice di sicurezza di Amazon EKS si trovano nelle Best practice per la sicurezza della Guida delle best practice di Amazon EKS.

## Analizzare le vulnerabilità in Amazon EKS

La sicurezza è una considerazione fondamentale per la configurazione e la manutenzione di cluster e applicazioni Kubernetes. Di seguito sono elencate le risorse per analizzare la configurazione di sicurezza dei cluster EKS, le risorse per verificare la presenza di vulnerabilità e le integrazioni con AWS i servizi che possono eseguire l'analisi al posto tuo.

# Il Center for Internet Security (CIS) Benchmark per Amazon EKS

Il Center for Internet Security (CIS) Benchmark fornisce indicazioni per le configurazioni di sicurezza di Amazon EKS. Il benchmark:

- È applicabile ai nodi Amazon EC2 (gestiti e autogestiti) in cui sei responsabile delle configurazioni di sicurezza dei componenti Kubernetes.
- Fornisce un modo standard approvato dalla community per garantire di aver configurato in modo sicuro il cluster Kubernetes e i nodi quando si utilizza Amazon EKS.
- È costituito da quattro sezioni: configurazione della registrazione del piano di controllo, configurazioni di sicurezza dei nodi, criteri e servizi gestiti.
- Supporta tutte le versioni Kubernetes attualmente disponibili in Amazon EKS e può essere eseguito utilizzando kube-bench, uno strumento open source standard per il controllo della configurazione utilizzando il benchmark CIS sui cluster Kubernetes.

Per ulteriori informazioni, consulta Presentazione del benchmark CIS Amazon EKS.

Per una `aws-sample` pipeline automatizzata per aggiornare il gruppo di nodi con un'AMI con benchmark CIS, consulta EKS-Optimized AMI Hardening Pipeline.

# Versioni della piattaforma Amazon EKS

Le versioni della piattaforma Amazon EKS rappresentano le funzionalità del piano di controllo del cluster, tra cui i flag del server API di Kubernetes abilitati e l'attuale versione della patch di Kubernetes. I nuovi cluster vengono implementati con la versione più recente della piattaforma. Per i dettagli, consulta EKS platform-versions.

È possibile aggiornare un cluster Amazon EKS alla versione più recente di Kubernetes. Man mano che nuove versioni di Kubernetes diventano disponibili in Amazon EKS, si consiglia di aggiornare tempestivamente i cluster in modo da usare la versione più recente disponibile. Per ulteriori informazioni sulle versioni di Kubernetes in EKS, consulta Amazon EKS supported versions.

# Elenco delle vulnerabilità del sistema operativo

## Elenco vulnerabilità di AL2023

Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux 2023 nel Centro di sicurezza di Amazon Linux o iscriviti al feed RSS associato. Gli eventi relativi a sicurezza e privacy includono

una panoramica del problema, dei pacchetti interessati e istruzioni per aggiornare le istanze per correggere il problema.

### Elenco delle vulnerabilità di Amazon Linux 2

Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux 2 nel Centro di sicurezza di Amazon Linux o iscriviti al feed RSS associato. Gli eventi relativi alla sicurezza e alla privacy includono una panoramica del problema, dei pacchetti interessati e istruzioni per aggiornare le istanze per correggere il problema.

## Rilevamento dei nodi con Amazon Inspector

È possibile utilizzare Amazon Inspector per verificare la presenza di accessibilità alla rete non intenzionale dei nodi e le vulnerabilità sulle istanze Amazon EC2.

## Rilevamento di cluster e nodi con Amazon GuardDuty

Amazon GuardDuty è un servizio di rilevamento delle minacce che aiuta a proteggere account, contenitori, carichi di lavoro e dati all'interno del tuo AWS ambiente. Tra le altre funzionalità, GuardDuty offre le seguenti due funzionalità che rilevano potenziali minacce ai cluster EKS: EKS Protection e Runtime Monitoring.

Per ulteriori informazioni, consulta the section called “Amazon GuardDuty”.

# Convalida della conformità per cluster di Amazon EKS

Per sapere se un AWS servizio rientra nell'ambito di specifici programmi di conformità, consulta AWS Services in Scope by Compliance Program e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di conformità di AWS.

Puoi scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download dei report in AWS Artifact.

La responsabilità dell'utente in materia di conformità nell'utilizzo dei AWS servizi è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per agevolare la conformità:

- Governance e conformità per la sicurezza: queste guide all'implementazione di soluzioni illustrano considerazioni relative all'architettura e i passaggi per implementare le funzionalità di sicurezza e conformità.

- Riferimenti sui servizi conformi ai requisiti HIPAA - Elenca i servizi HIPAA idonei. Non tutti i AWS servizi sono idonei all'HIPAA.
- AWS Risorse sulla conformità: questa raccolta di cartelle di lavoro e guide potrebbe essere applicabile al tuo settore e alla tua sede.
- AWS Guide alla conformità dei clienti: comprendi il modello di responsabilità condivisa dal punto di vista della conformità. Le guide riassumono le migliori pratiche per proteggere AWS i servizi e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- Valutazione delle risorse con regole nella AWS Config Developer Guide: il servizio Config valuta in che misura le AWS configurazioni delle risorse sono conformi alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub: questo AWS servizio fornisce una visione completa dello stato di sicurezza interno. AWS Security Hub utilizza i controlli di sicurezza per valutare le AWS risorse e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- Amazon GuardDuty: questo AWS servizio rileva potenziali minacce ai tuoi AWS account, carichi di lavoro, contenitori e dati monitorando l'ambiente per individuare attività sospette e dannose. GuardDuty può aiutarti a soddisfare vari requisiti di conformità, come PCI DSS, soddisfacendo i requisiti di rilevamento delle intrusioni richiesti da determinati framework di conformità.
- AWS Audit Manager: questo AWS servizio consente di verificare continuamente AWS l'utilizzo per semplificare la gestione del rischio e la conformità alle normative e agli standard di settore.

# Considerazioni di sicurezza per Amazon Elastic Container Service per Kubernetes

Di seguito sono riportate alcune considerazioni relative alla sicurezza del cloud, in quanto influiscono su Amazon EKS.

Argomenti

- Sicurezza dell'infrastruttura in Amazon EKS
- Comprendere la resilienza nei cluster Amazon EKS
- Prevenzione del problema "confused deputy" tra servizi in Amazon EKS

# Sicurezza dell'infrastruttura in Amazon EKS

In quanto servizio gestito, Amazon Elastic Kubernetes Service è protetto dalla sicurezza di rete globale di AWS. Per informazioni sui servizi di sicurezza AWSe su come AWSprotegge l'infrastruttura, consulta la pagina Sicurezza del cloud AWS. Per progettare l'ambiente AWSutilizzando le best practice per la sicurezza dell'infrastruttura, consulta la pagina Protezione dell'infrastruttura nel Pilastro della sicurezza di AWSWell-Architected Framework.

Utilizzare le chiamate API pubblicate di AWS per accedere a Amazon EKS tramite la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, puoi utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per firmare le richieste.

Quando si crea un cluster Amazon EKS, specificare anche le sottoreti VPC; che devono essere utilizzate dal cluster. Amazon EKS richiede sottoreti in almeno due zone di disponibilità. È consigliabile un VPC con sottoreti pubbliche e private in modo che Kubernetes possa creare bilanciatori del carico pubblici nelle sottoreti pubbliche, che bilancino il carico del traffico ai pod in esecuzione su nodi che si trovano in sottoreti private.

Per ulteriori informazioni sulle considerazioni per il VPC, consultare the section called "Requisiti del VPC e delle sottoreti".

Se si crea il proprio VPC e i gruppi di nodi con i modelli AWS CloudFormation forniti nella procedura dettagliata Get started with Amazon EKS, il piano di controllo e i gruppi di sicurezza dei nodi di lavoro vengono configurati con le impostazioni consigliate.

Per ulteriori informazioni sulle considerazioni per i gruppi di sicurezza, consultare the section called "Requisiti relativi al gruppo di sicurezza".

Quando si crea un nuovo cluster, Amazon EKS crea un endpoint per il server API Kubernetes gestito utilizzato per comunicare con il cluster (usando strumenti di gestione Kubernetes, ad esempio `kubectl`). Per impostazione predefinita, questo endpoint del server API è pubblico in Internet

e l'accesso al server API è protetto utilizzando una combinazione di AWS Identity and Access Management (AWS IAM) e dal controllo degli accessi basato sul ruolo (RBAC) Kubernetes nativo.

Puoi abilitare l'accesso privato al server API Kubernetes in modo che tutte le comunicazioni tra i nodi di lavoro e il server API rimangano all'interno del VPC. È possibile limitare gli indirizzi IP che possono accedere al server API da Internet o disabilitare completamente l'accesso a Internet al server API.

Per ulteriori informazioni sulla modifica dell'accesso all'endpoint del cluster, consultare the section called "Modifica dell'accesso all'endpoint del cluster".

È possibile implementare policy di rete Kubernetes con CNI di Amazon VPC o strumenti di terze parti come Project Calico. Per ulteriori informazioni sull'utilizzo della CNI di Amazon VPC per le policy di rete, consulta la pagina the section called "Policy di Kubernetes". Project Calico è un progetto open source di terze parti. Per ulteriori informazioni, consulta la documentazione di Project Calico.

## Accedi ad Amazon EKS utilizzando AWS PrivateLink

Puoi utilizzarlo AWS PrivateLink per creare una connessione privata tra il tuo VPC e Amazon Elastic Kubernetes Service. Puoi accedere ad Amazon EKS come se fosse nel tuo VPC, senza l'uso di un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione Direct Connect. AWS Le istanze nel tuo VPC non richiedono indirizzi IP pubblici per l'accesso ad Amazon EKS.

Stabilisci questa connessione privata creando un endpoint di interfaccia alimentato da AWS PrivateLink. In ciascuna sottorete viene creata un'interfaccia di rete endpoint da abilitare per l'endpoint di interfaccia. Queste sono interfacce di rete gestite dal richiedente che fungono da punto di ingresso per il traffico destinato ad Amazon EKS.

Per ulteriori informazioni, consulta Access AWS services through AWS PrivateLink nella AWS PrivateLink Guida.

Prima di iniziare

Prima di iniziare, esegui le seguenti attività:

- Leggi Access and AWS service using an interface (VPC endpoint) nella Guida. AWS PrivateLink

Considerazioni

- Supporto e limitazioni: gli endpoint dell'interfaccia Amazon EKS consentono l'accesso sicuro a tutte le azioni dell'API Amazon EKS dal tuo VPC, ma presentano limitazioni specifiche, ovvero non supportano l'accesso alle API Kubernetes, poiché dispongono di un endpoint privato separato

né è possibile configurare Amazon EKS in modo che sia accessibile solo tramite l'endpoint di interfaccia. L'endpoint di rilevamento OIDC e JWKS del cluster utilizzato dai ruoli IAM per gli account di servizio (IRSA) viene inoltre gestito da un endpoint di interfaccia separato con il proprio nome di servizio e un comportamento di controllo degli accessi; vedere. the section called "Accedi all'endpoint OIDC del cluster utilizzando AWS PrivateLink"

- Prezzi: l'utilizzo degli endpoint di interfaccia per Amazon EKS comporta AWS PrivateLink costi standard: tariffe orarie per ogni endpoint fornito in ciascuna zona di disponibilità, costi di elaborazione dei dati per il traffico attraverso l'endpoint. Per ulteriori informazioni, consulta Prezzi di AWS PrivateLink .
- Sicurezza e controllo degli accessi: consigliamo di migliorare la sicurezza e controllare l'accesso con queste configurazioni aggiuntive: usa le policy degli endpoint VPC per controllare l'accesso ad Amazon EKS tramite l'endpoint dell'interfaccia, associa i gruppi di sicurezza alle interfacce di rete degli endpoint per gestire il traffico, usa i log di flusso VPC per acquisire e monitorare il traffico IP da e verso gli endpoint dell'interfaccia, con registri pubblicabili su Amazon CloudWatch o Amazon S3. Per ulteriori informazioni, consulta Controllare l'accesso agli endpoint VPC utilizzando le policy degli endpoint e Registrare il traffico IP utilizzando log di flusso VPC.
- Opzioni di connettività: gli endpoint di interfaccia offrono opzioni di connettività flessibili utilizzando l'accesso locale (connetti il tuo data center locale a un VPC con l'endpoint di interfaccia utilizzando AWS Direct Connect o AWS Site-to-Site VPN) o tramite la connettività inter-VPC (usa AWS Transit Gateway o il peering VPC per connettere altri VPC al VPC con l'endpoint dell'interfaccia, mantenendo il traffico all'interno della AWS rete).
- Supporto versione IP: gli endpoint creati prima di agosto 2024 supportano solo IPv4 utilizzando eks.region.amazonaws.com. I nuovi endpoint creati dopo agosto 2024 supportano IPv4 e IPv6 dual-stack (ad esempio, eks.region.amazonaws.com, eks.region.api.aws).
- Disponibilità regionale: AWS PrivateLink per l'API EKS non è disponibile nelle regioni Asia Pacifico (Malesia) (ap-southeast-5), Asia Pacifico (Thailandia) (ap-southeast-7), Messico (Centrale) (mx-central-1) e Asia Pacifico (Taipei) (ap-east-2). AWS PrivateLink il supporto per eks-auth (EKS Pod Identity) è disponibile nella regione Asia Pacifico (Malesia) (ap-southeast-5). AWS PrivateLink per il cluster OIDC endpoint () è disponibile nella maggior parte delle regioni. `com.amazonaws.region-code.oidc-eks` AWS

### Crea un endpoint di interfaccia per Amazon EKS

Puoi creare un endpoint di interfaccia per Amazon EKS utilizzando la console Amazon VPC o l'interfaccia a riga di AWS comando (CLI).AWS Per ulteriori informazioni, consulta Creare un endpoint VPC nella Guida. AWS PrivateLink

Crea un endpoint di interfaccia per Amazon EKS utilizzando i seguenti nomi servizio:

API EKS

- com.amazonaws.region-code.eks
- com.amazonaws.region-code.eks-fips (per gli endpoint) FIPS-compliant

API di autenticazione EKS (Pod Identity EKS)

- com.amazonaws.region-code.eks-auth

Endpoint OIDC del cluster EKS (IRSA discover/JWKS)

- com.amazonaws.region-code.oidc-eks

Questo endpoint ha un comportamento di controllo degli accessi diverso dagli endpoint dell'API EKS. Per ulteriori informazioni, consulta the section called "Accedi all'endpoint OIDC del cluster utilizzando AWS PrivateLink".

Visualizzazioni delle risorse della console EKS

- com.amazonaws.region-code.eks-proxy

Questo servizio endpoint supporta le visualizzazioni delle risorse del cluster nelle console. AWS Viene utilizzato da console e servizi AWS gestiti, non viene richiamato direttamente dalle applicazioni (non esistono SDK o CLI pubblici). Crea questo endpoint quando il tuo VPC non ha un'uscita Internet o quando sostituisci la zona in una zona ospitata privata. `eks.region-code.amazonaws.com` In questi casi è necessario per il funzionamento delle visualizzazioni delle risorse della console.

Questo endpoint registra due nomi DNS privati:

- `eks-proxy.region-code.api.aws`(dual-stack)
- `eks-proxy.eks.region-code.amazonaws.com`(IPv4; utilizzalo sulle reti) IPv4-only

Funzionalità DNS privata per gli endpoint dell'interfaccia Amazon EKS

La funzionalità DNS privato, abilitata di default per gli endpoint di interfaccia di Amazon EKS e altri AWS servizi, facilita le richieste API sicure e private utilizzando nomi DNS regionali predefiniti.

Questa funzionalità garantisce che le chiamate API vengano indirizzate attraverso l'endpoint dell'interfaccia sulla AWS rete privata, migliorando la sicurezza e le prestazioni.

La funzionalità DNS privato si attiva automaticamente quando si crea un endpoint di interfaccia per Amazon EKS o altri servizi. AWS Per abilitarla, devi configurare correttamente il tuo VPC impostando attributi specifici:

- abilita DnsHostnames: consente alle istanze all'interno del VPC di avere nomi host DNS.
- abilita DnsSupport: abilita la risoluzione DNS in tutto il VPC.

Per istruzioni dettagliate per verificare o modificare queste impostazioni, consulta Visualizzare e aggiornare gli attributi DNS per il VPC.

Nomi DNS e tipi di indirizzi IP

Con la funzionalità DNS privata abilitata, è possibile utilizzare nomi DNS specifici per connetterti ad Amazon EKS e queste opzioni si evolvono nel tempo:

- eks.region.amazonaws.com: il nome DNS tradizionale, che si risolve solo negli indirizzi IPv4 prima di agosto 2024. Per gli endpoint esistenti aggiornati al dual-stack, questo nome è risolto sia in indirizzi IPv4 che IPv6.
- eks.region.api.aws: disponibile per i nuovi endpoint creati dopo agosto 2024, questo nome DNS dual-stack si risolve in indirizzi IPv4 e IPv6.

Dopo agosto 2024, i nuovi endpoint di interfaccia sono forniti con due nomi DNS ed è possibile optare per il tipo di indirizzo IP dual-stack. Per gli endpoint esistenti, l'aggiornamento a dual-stack modifica eks.region.amazonaws.com in modo che supporti sia IPv4 che IPv6.

Utilizzo della funzionalità DNS privata

Una volta configurata, la funzionalità DNS privata può essere integrata nei flussi di lavoro, offrendo le seguenti funzionalità:

- Richieste API: utilizza i nomi DNS regionali predefiniti, `eks.region.amazonaws.com` o `eks.region.api.aws`, in base alla configurazione dell'endpoint per effettuare richieste API ad Amazon EKS.
- Compatibilità delle applicazioni: le applicazioni esistenti che chiamano le API EKS non richiedono modifiche per sfruttare questa funzionalità.

- AWS CLI con Dual-Stack: per utilizzare gli endpoint dual-stack con la AWS CLI, consulta la configurazione degli endpoint Dual-stack e FIPS nella Guida di riferimento di SDK and Tools. AWS
- Routing automatico: qualsiasi chiamata all'endpoint del servizio predefinito di Amazon EKS è instradata automaticamente tramite l'endpoint di interfaccia, garantendo una connettività privata e sicura.

Accedi all'endpoint OIDC del cluster utilizzando AWS PrivateLink

Ogni cluster Amazon EKS pubblica un documento di rilevamento OpenID Connect (OIDC) pubblico e un JSON Web Key Set (JWKS) all'URL dell'emittente OIDC. I ruoli IAM per gli account di servizio (IRSA) utilizzano questo endpoint per pubblicare le chiavi di firma pubbliche del cluster. Gli strumenti che configurano l'IRSA o convalidano i token recuperano direttamente le chiavi da questo endpoint.

Puoi utilizzarlo AWS PrivateLink per raggiungere questo endpoint OIDC privatamente dal tuo VPC, senza uscita da Internet. Ciò è utile per i VPC privati o senza uscita in cui gli strumenti in-VPC devono raggiungere l'endpoint OIDC, ad esempio creando il provider di identità IAM OIDC del cluster (`eksctl`, Terraform o CLI) o eseguendo i propri validatori di token all'interno del VPC. AWS In precedenza, il raggiungimento di questo endpoint richiedeva l'accesso a Internet o soluzioni alternative al DNS (vedi containers-roadmap #2038 su). https://github.com/aws/containers-roadmap/issues/2038 GitHub

Crea un endpoint di interfaccia per l'endpoint OIDC del cluster utilizzando il seguente nome di servizio:

- `com.amazonaws.region-code.oidc-eks`

Considerazioni

- Questo endpoint non supporta le policy degli endpoint VPC. I documenti OIDC discovery e JWKS vengono forniti in forma anonima. Ciò è richiesto dalla specifica OpenID Connect ed è il modo in cui funziona IRSA: i validatori recuperano le chiavi pubbliche senza presentare le credenziali. Poiché le richieste non comportano alcun principio o azione IAM, una policy sugli endpoint VPC non ha nulla da confrontare. Questo endpoint accetta solo la policy di accesso completo predefinita. Controlla chi può raggiungere l'endpoint con i gruppi di sicurezza sulle interfacce di rete degli endpoint e con il routing della sottorete.

- La connettività privata non è un limite di autorizzazione.  Il raggiungimento dell'endpoint OIDC AWS PrivateLink mantiene questo traffico sulla AWS rete, ma non controlla di per sé chi può assumere un ruolo IAM. I dati forniti sono materiali a chiave pubblica. La politica di fiducia del ruolo assunto impone l'autorizzazione IRSA, non il percorso di rete utilizzato per raggiungere l'endpoint OIDC. Questa applicazione avviene quando un carico di lavoro chiama. `sts:AssumeRoleWithWebIdentity` Controlla quali carichi di lavoro possono assumere un ruolo con `sub` le condizioni `aud` e nella politica di fiducia del ruolo; vedi Assegnare ruoli IAM agli account di servizio Kubernetes.
- AWS STS convalida i token seguendo il proprio percorso.  Questo endpoint non influisce sul modo in cui AWS STS convalida i token IRSA. Quando un carico di lavoro chiama`AssumeRoleWithWebIdentity`, STS recupera i JWKS del cluster dall'interno AWS, non tramite il tuo VPC o questo endpoint. La creazione di questo endpoint o l'override del DNS OIDC nel VPC non modifica la convalida dei token. Per mantenere privata anche la `AssumeRoleWithWebIdentity` chiamata del carico di lavoro, create un endpoint di interfaccia AWS STS e configurate separatamente l'endpoint STS regionale; consultate Configurare l'endpoint Security Token Service per un account di servizio. AWS
- Prezzi:  si applicano le tariffe standard. AWS PrivateLink Vedi AWS PrivateLink i prezzi.

DNS privati e nomi dual-stack

Ogni cluster ha un nome host dell'emittente OIDC, determinato dalla famiglia IP del cluster. I cluster IPv4, che sono i valori predefiniti, utilizzano. `oidc.eks.region-code.amazonaws.com` I cluster IPv6 utilizzano il nome host `oidc-eks.region-code.api.aws` dual-stack, che si risolve in indirizzi IPv4 e IPv6. Nelle regioni cinesi AWS , il nome host dual-stack è. `oidc-eks.region-code.api.amazonwebservices.com.cn`

L'endpoint dell'interfaccia registra entrambi i nomi host. Con il DNS privato abilitato (impostazione predefinita), ogni nome host si risolve nell'endpoint dell'interfaccia. L'URL emittente del cluster si risolve quindi nell'endpoint indipendentemente dal fatto che il cluster utilizzi il nome host o il nome host dual-stack. `oidc.eks.region-code.amazonaws.com` Un client che segue l'URL dell'emittente di un token per recuperare il documento di scoperta e JWKS raggiunge l'endpoint privatamente in entrambi i casi.

## Comprendere la resilienza nei cluster Amazon EKS

L'infrastruttura AWS globale è costruita attorno AWS a regioni e zone di disponibilità. AWS Le regioni forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza,

ad alto throughput e altamente ridondanti. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Amazon EKS esegue e ridimensiona il piano di controllo Kubernetes su più zone di disponibilità per garantire un'elevata AWS disponibilità. Amazon EKS dimensiona automaticamente le istanze del piano di controllo in base al carico, rileva e sostituisce le istanze del piano di controllo non integre e fornisce loro le patch automaticamente. Dopo aver avviato un aggiornamento della versione, Amazon EKS aggiorna il piano di controllo per conto dell'utente, mantenendone un'elevata disponibilità durante l'aggiornamento.

Questo piano di controllo è composto da almeno due istanze di server API e tre `etcd` istanze eseguite su tre zone di disponibilità all'interno di una regione. AWS Amazon EKS:

- Monitora attivamente il carico sulle istanze del piano di controllo e le ridimensiona automaticamente per garantire prestazioni elevate.
- Rileva e sostituisce automaticamente le istanze del piano di controllo non funzionanti, riavviandole tra le zone di disponibilità all'interno della regione in base alle esigenze. AWS
- Sfrutta l'architettura delle AWS regioni per mantenere un'elevata disponibilità. Per questo motivo, Amazon EKS è in grado di offrire un SLA per la disponibilità dell'endpoint del server API.

Per ulteriori informazioni su AWS regioni e zone di disponibilità, consulta infrastruttura AWS globale.

## Prevenzione del problema "confused deputy" tra servizi in Amazon EKS

Con "confused deputy" si intende un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire una certa operazione può costringere un'entità con più privilegi a eseguire tale operazione. Nel AWS, l'impersonificazione tra servizi può portare al confuso problema del vice. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare che ciò accada, AWS mette a disposizione strumenti che consentono di proteggere i dati relativi a tutti i servizi, con responsabili del servizio a cui è stato concesso l'accesso alle risorse del vostro account.

Consigliamo di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy delle risorse per limitare le autorizzazioni con cui Amazon Elastic Kubernetes Service (Amazon EKS) fornisce un altro servizio alla risorsa.

`aws:SourceArn`

Utilizza `aws:SourceArn` per associare una sola risorsa all'accesso tra servizi.

`aws:SourceAccount`

Utilizza `aws:SourceAccount` se desideri consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Se l'ARN completo della risorsa non è noto o si scelgono più risorse, utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:<servicename>:*:<123456789012>:*`.

Se il valore `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare sia `aws:SourceAccount` che `aws:SourceArn` per limitare le autorizzazioni.

## Prevenzione del problema “confused deputy” tra servizi nei cluster Amazon EKS

# richiesto un ruolo IAM del cluster Amazon EKS per ogni cluster. I cluster Kubernetes gestiti da Amazon EKS utilizzano questo ruolo per gestire i nodi e il provider cloud legacy utilizza questo ruolo per creare bilanciatori del carico con Elastic Load Balancing per i servizi. Queste operazioni del cluster possono influire solo sullo stesso account, quindi consigliamo di limitare ogni ruolo del cluster a quel cluster e a quell'account. Questa è un'applicazione specifica della AWS raccomandazione di seguire il principio del privilegio minimo nel proprio account.

Formato dell'ARN dell'origine

Il valore di `aws:SourceArn` deve essere l'ARN di un cluster EKS nel formato `arn:aws:eks:`*`region`*`:`*`account`*`:cluster/`*`cluster-name`*. Ad esempio, `arn:aws: eks:us-west-2:123456789012:cluster/my-cluster`.

Formato della policy di attendibilità per i ruoli del cluster EKS

L'esempio seguente mostra il modo in cui è possibile utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` delle condizioni globali in Amazon EKS per prevenire il problema "confused deputy".

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:eks:us-west-2:123456789012:cluster/my-cluster"
          },
        "StringEquals": {
            "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

# Considerazioni sulla sicurezza per Kubernetes

Di seguito sono riportate alcune considerazioni relative alla sicurezza del cloud, in quanto influiscono su Kubernetes nei cluster Amazon EKS. Per un'analisi approfondita dei controlli e delle pratiche di sicurezza in Kubernetes, consulta Sicurezza nativa del cloud e Kubernetes nella documentazione di Kubernetes.

Argomenti

- Proteggere i carichi di lavoro con i certificati Kubernetes
- Ruota l'autorità di certificazione (CA) del cluster EKS
- Comprendere ruoli e utenti creati da Amazon EKS tramite RBAC
- Crittografia dei segreti Kubernetes con KMS su cluster esistenti
- Utilizzare i segreti di AWS Secrets Manager con i pod di Amazon EKS

- Crittografia a busta predefinita per tutti i dati dell'API Kubernetes
- Harden Kubernetes RBAC in Amazon EKS

## Proteggere i carichi di lavoro con i certificati Kubernetes

L'API Kubernetes Certificates automatizza il provisioning delle credenziali. X.509 L'API presenta un'interfaccia a riga di comando che consente ai client API Kubernetes di richiedere e ottenere X.509 certificati da un'autorità di certificazione (CA). Puoi utilizzare la risorsa (CSR) `CertificateSigningRequest` per chiedere la firma del certificato da parte di un firmatario designato. Le richieste vengono approvate o negate prima della firma. Kubernetes supporta sia i firmatari integrati che personalizzati con comportamenti ben definiti. In questo modo, i client possono prevedere cosa succede alle loro CSR. Per ulteriori informazioni sulla firma dei certificati, consulta signing requests.

Uno dei firmatari integrati è `kubernetes.io/legacy-unknown`. L'API `v1beta1` della risorsa CSR ha onorato questo firmatario "legacy-unknown". Tuttavia, l'API `v1` stabile di CSR non consente l'impostazione di `signerName` su `kubernetes.io/legacy-unknown`.

Se desideri utilizzare l'autorità di certificazione (CA) di Amazon EKS per la generazione di certificati sul tuo cluster, devi utilizzare un firmatario personalizzato. Per utilizzare la versione `v1` dell'API CSR e generare un nuovo certificato, devi eseguire la migrazione di tutti i manifesti e i client dell'API esistenti. I certificati esistenti creati con l'API `v1beta1` esistente sono validi e funzionanti fino alla scadenza dei certificati. Questo include gli output seguenti:

- Distribuzione di attendibilità: nessuna. Non esiste alcuna distribuzione di attendibilità o standard per questo firmatario in un cluster Kubernetes.
- Argomenti consentiti: tutti
- Estensioni x509 consentite: rispetta le estensioni di utilizzo dell'oggetto AltName e della chiave e scarta le altre estensioni
- Utilizzo delle chiavi consentite: non devono includere altri utilizzi oltre ["key encipherment", "digital signature", "server auth"]

> **Note**
>
> La firma dei certificati client non è supportata.

- Expiration/certificate durata: 45 giorni (impostazione predefinita e massima)

- CA bit allowed/disallowed: non consentito

## Generazione CSR di esempio con signerName

Questi passaggi mostrano come generare un certificato di servizio per l'`myserver.default.svc`utilizzo `signerName: beta.eks.amazonaws.com/app-serving` di nomi DNS. Utilizzala come guida per il tuo ambiente.

1. Esegui il comando `openssl genrsa -out myserver.key 2048` per generare una chiave privata RSA.

```
openssl genrsa -out myserver.key 2048
```

2. Utilizza il comando seguente per generare una richiesta di certificato.

```
openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
```

3. Genera un valore `base64` per la richiesta CSR e memorizzalo in una variabile da utilizzare in un passaggio successivo.

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "
")
```

4. Per creare un file denominato `mycsr.yaml`, esegui il comando di seguito. Nell'esempio seguente, `beta.eks.amazonaws.com/app-serving` è `signerName`.

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. Invia la CSR.

```
kubectl apply -f mycsr.yaml
```

6. Approva il certificato di servizio.

```
kubectl certificate approve myserver
```

7. Verifica l'emissione del certificato.

```
kubectl get csr myserver
```

Di seguito viene riportato un output di esempio.

```
NAME       AGE     SIGNERNAME                              REQUESTOR          CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. Esporta il certificato emesso.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d >
 myserver.crt
```

## Ruota l'autorità di certificazione (CA) del cluster EKS

Nell'infrastruttura a chiave pubblica (PKI), un'autorità di certificazione (CA) è un'entità attendibile che emette e firma certificati digitali. Questi certificati stabiliscono l'identità e consentono la comunicazione crittografata tra sistemi tramite TLS (Transport Layer Security). Quando un client si connette a un server, il server presenta un certificato firmato da una CA. Il client verifica il certificato del server rispetto alle CA di cui si fida prima di consentire il proseguimento della connessione.

In Amazon Elastic Kubernetes Service (Amazon EKS), viene creata una CA per ogni cluster EKS al momento della creazione del cluster. Segue lo stesso modello di Kubernetes upstream: ogni cluster EKS ha una propria CA che firma i certificati per il server API. Questo è ciò che consente ai componenti del piano di controllo, ai nodi di lavoro e ai client di autenticarsi sul server API e stabilire connessioni crittografate al cluster EKS.

Queste autorità di certificazione (CA) hanno un periodo di validità definito. La rotazione delle CA è il processo di sostituzione dell'autorità di certificazione del cluster EKS prima della scadenza, garantendo che il cluster rimanga operativo e accessibile. Disponiamo di protezioni integrate

che gestiscono automaticamente questo processo. Se non avvii tu stesso la rotazione delle CA, aggiungeremo automaticamente una CA successore e la attiveremo prima della scadenza della CA in uscita, assicurando che il cluster rimanga disponibile.

Durante la rotazione delle CA, una CA successore viene aggiunta al cluster EKS. EKS distribuisce automaticamente la CA successore a tutti i componenti AWS gestiti (piano di controllo, istanze EKS Auto Mode e AWS nodi Fargate). Sei responsabile dell'aggiornamento dei nodi di lavoro che gestisci (non EKS Auto Mode, non Fargate) e dei client esterni (come i file e le CI/CD pipeline kubeconfig) in modo che considerino attendibile la CA successore prima che venga attivata. Dopo l'attivazione della CA successore, il cluster EKS passa alla firma dei certificati con la CA successore. La CA in uscita viene quindi ritirata.

La rotazione delle CA è richiesta per ogni cluster EKS perché le CA hanno un periodo di validità finito. Il periodo di validità dipende da quando è stata creata la CA del cluster. Per maggiori dettagli, consulta la sezione delle domande frequenti. Le protezioni EKS garantiscono che il cluster stesso rimanga disponibile per tutto il ciclo di vita della rotazione, ma il successo della rotazione dipende anche dall'aggiornamento dei nodi di lavoro che gestisci e dei client esterni in modo che mantengano la connettività dopo l'attivazione della CA successore.

Le API, l'esperienza della console, le notifiche e le linee guida dettagliate nelle sezioni che seguono sono progettate per supportarti in questo processo. L'intero ambito delle responsabilità è trattato nella sezione relativa al modello di responsabilità condivisa.

## Come funziona la rotazione di CA

La rotazione della CA in Amazon EKS è un processo in più fasi. Disponiamo di misure di sicurezza automatiche per preservare la disponibilità dei cluster EKS durante l'intero ciclo di vita della rotazione, indipendentemente dal fatto che tu agisca o meno. Una rotazione riuscita, in cui tutti i componenti mantengano la connettività, richiede i passaggi descritti nelle sezioni seguenti.

Fase 1: aggiungere una CA successore

Una CA successore viene aggiunta al cluster EKS. Da questo punto, il cluster EKS considera attendibili contemporaneamente sia la CA in uscita che la CA successore. I certificati continuano ad essere emessi dalla CA uscente. Non si verificano interruzioni.

Puoi aggiungere tu stesso una CA successore in qualsiasi momento utilizzando la AWS CLI, l'API EKS, la Console o l'infrastruttura come codice (IaC), ad esempio AWS CloudFormation, purché il cluster sia attivo. Se non lo fai, ne aggiungeremo automaticamente uno per tuo conto.

La CA successore non può essere attivata fino al AWS completamento della distribuzione ai componenti AWS gestiti in EKS (Fase 2). Questo è il momento giusto per iniziare a identificare i nodi di lavoro e i client esterni che devono essere aggiornati. L'identificazione di tutti i sistemi che si connettono al server API del cluster EKS può richiedere tempo, in particolare in ambienti con più team, CI/CD pipeline e strumenti di monitoraggio. L'avvio anticipato di questo processo offre la massima flessibilità per coordinare gli aggiornamenti in base alle proprie tempistiche.

Fase 2: distribuzione della CA successiva

Dopo aver aggiunto la CA successore, AWS aggiorna i componenti gestiti nel cluster EKS (il piano di controllo, le istanze EKS Auto Mode e i nodi AWS Fargate) per riconoscere e considerare attendibili entrambe le CA. È possibile monitorarne l'avanzamento tramite lo stato di distribuzione della CA. La CA successore non può essere attivata finché la distribuzione non è completa.

Dopo aver completato la distribuzione della CA ai componenti gestiti, sei responsabile dell'aggiornamento di due gruppi: i nodi di lavoro che gestisci (modalità automatica non EKS, non Fargate) e i client esterni affinché si fidino della CA successore. Ciò garantisce che continueranno a connettersi al server API dopo l'attivazione della CA successore. Se alcuni componenti non vengono aggiornati prima dell'attivazione della CA successore, è disponibile il rollback della CA per ripristinare la connettività mentre si completano gli aggiornamenti rimanenti.

Fase 3: attivazione della CA successore

Dopo aver completato la distribuzione della CA successore a tutti i componenti AWS gestiti in EKS, è possibile attivare la CA successore. Ti consigliamo di attivare la CA successore secondo la tua tempistica. Concediti il tempo necessario per scoprire e aggiornare i nodi di lavoro che gestisci e i client esterni per fidarti della CA successore. Dopo l'attivazione della CA successore, il cluster EKS emette i certificati firmati dalla CA successore. La CA in uscita rimane affidabile ma non viene più utilizzata per la firma. Una finestra di rollback è disponibile per un periodo limitato dopo l'attivazione, che consente di tornare alla CA in uscita, se necessario. Il rollback è trattato in dettaglio in una sezione successiva.

Puoi attivare tu stesso la CA successore quando sei sicuro che i nodi di lavoro che gestisci e i client siano stati aggiornati. In caso contrario, attiveremo automaticamente la CA successore all'avvicinarsi della scadenza.

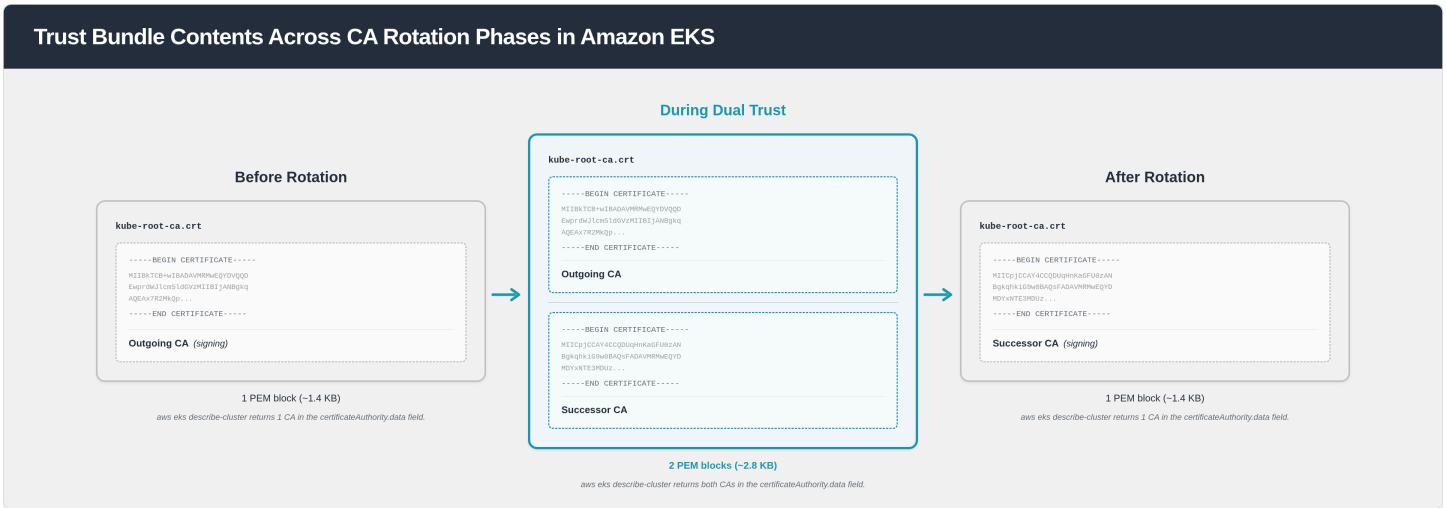
Il periodo di doppia fiducia

Il periodo che intercorre tra l'aggiunta di una CA successore (Fase 1) e il ritiro della CA uscente è chiamato periodo di doppio trust. Durante questo periodo, il cluster EKS considera attendibili

entrambe le CA contemporaneamente. Questo è ciò che rende la rotazione senza interruzioni: i componenti possono essere aggiornati in modo incrementale perché il cluster EKS accetta i certificati firmati da entrambe le CA.

Il periodo di doppia fiducia ti dà il tempo di identificare e aggiornare tutti i nodi di lavoro che gestisci e i client esterni senza dover coordinare tutte le modifiche contemporaneamente.

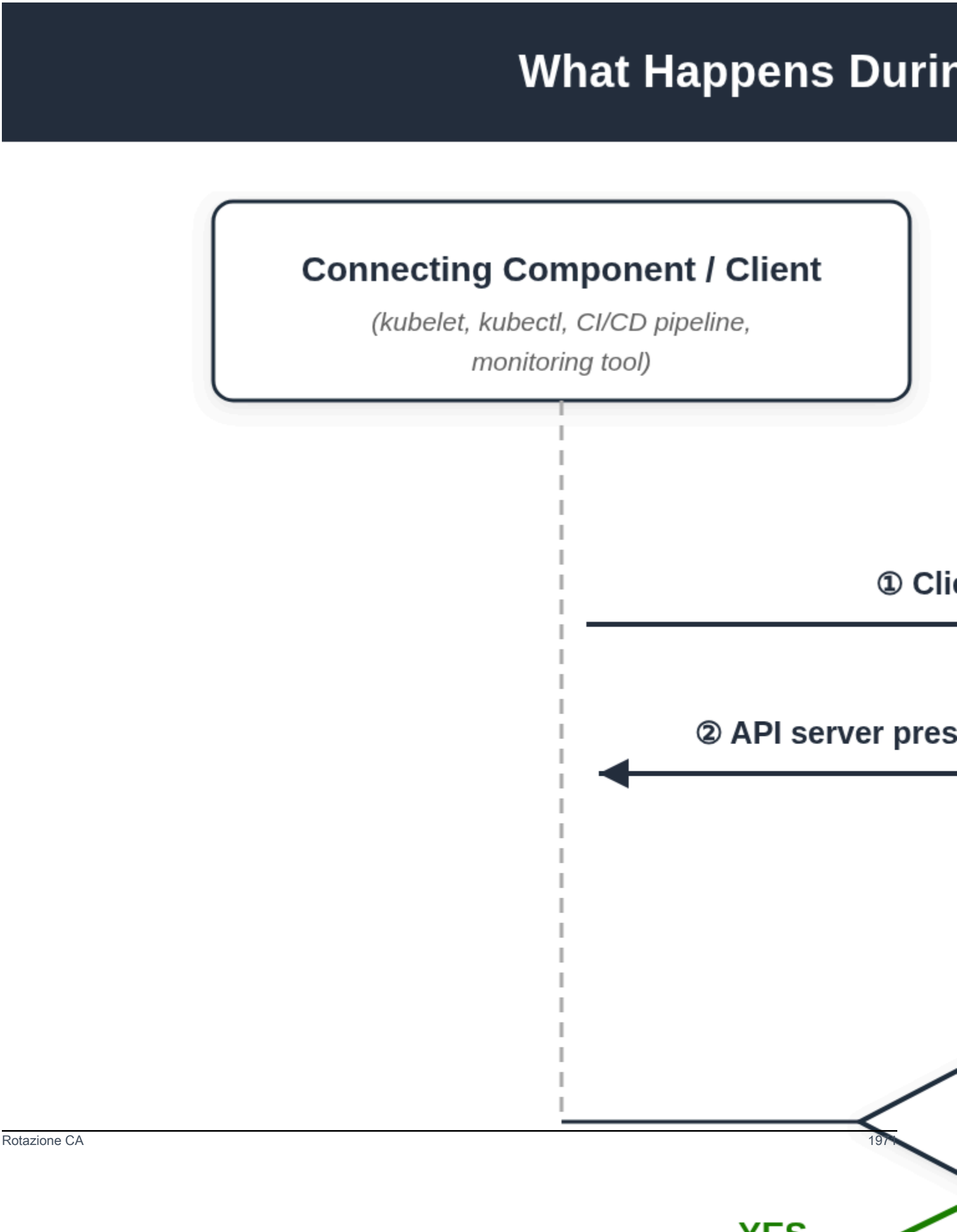
> **Note**
>
> Durante il periodo di doppia fiducia, il pacchetto di fiducia del cluster contiene due CA. Questo è un comportamento standard per i pacchetti di fiducia con codifica .pem. Aggiorna le applicazioni che eseguono una rigorosa convalida a CA singola o il pinning CA per accettare più CA prima dell'inizio della rotazione.




È importante capire in che modo l'attivazione della CA successore influisce sulla connettività. Quando un client si connette al server API, verifica l'identità del server controllando che il certificato del server sia stato firmato da una CA di cui si fida.

Dopo l'attivazione della CA successore, il server API presenta il certificato firmato dalla CA successore. I client che hanno aggiornato il trust bundle per includere la CA successore eseguiranno la verifica con successo e si connetteranno normalmente. I client che non hanno aggiornato il trust bundle non riconosceranno il certificato del server e non saranno in grado di stabilire una connessione.

Il diagramma seguente mostra il flusso di connessione TLS dopo l'attivazione della CA successiva.


What Happens Durin
Connecting Component / Client
(kubelet, kubectl, CI/CD pipeline, monitoring tool)
① Cli
② API server pres
YES

Ecco perché è importante aggiornare i nodi di lavoro e i client esterni prima dell'attivazione della CA successore: hanno bisogno della CA successore nel loro trust bundle per verificare l'identità del server API e connettersi. Il periodo di doppia fiducia e il rollback di CA forniscono tempo e una rete di sicurezza per completare questa operazione.

# Amazon EKS CA Rotation

## Stage 1 — Append CA

### WHAT HAPPENS

Successor CA added to the cluster. Cluster now tru
CAs. Certificates still issued by outgoing CA.

### WHO ACTS

*You or AWS (auto-append at T−2 years before CA*

## Modello di responsabilità condivisa

La rotazione delle CA in Amazon EKS segue lo stesso modello di responsabilità condivisa che si applica in generale a tutti. AWS AWS è responsabile della sicurezza e della disponibilità dell'infrastruttura cloud e tu sei responsabile della sicurezza e della configurazione dei tuoi carichi di lavoro al suo interno. Per ulteriori informazioni su come la responsabilità condivisa si applica ad Amazon EKS, consulta le best practice di sicurezza di EKS.

Nel contesto della rotazione delle CA, ciò significa:

AWS è responsabile per

- Aggiornamento del piano di controllo del cluster EKS in modo che consideri attendibile ed emetta i certificati dalla CA successore
- Aggiornamento dei nodi EKS Auto Mode in modo che ritengano attendibile la CA successore
- Aggiornamento dei nodi AWS Fargate in modo che ritengano attendibile la CA successore
- Garantire che la CA successore non possa essere attivata fino al completamento della distribuzione ai componenti AWS gestiti in EKS
- Preservazione della disponibilità del cluster EKS per tutto il ciclo di vita della rotazione
- Inviando una notifica in ogni fase del processo di rotazione
- Avvio automatico della rotazione se non avete agito prima che la CA si avvicini alla scadenza

Sei responsabile per

- Aggiornamento dei clienti esterni (workstation per sviluppatori, CI/CD pipeline, strumenti di monitoraggio, automazione) in modo che si affidino alla CA successore
- Aggiornamento dei nodi di lavoro (gruppi di nodi gestiti, Karpenter-controlled nodi, nodi autogestiti, nodi ibridi) per rendere attendibile la CA successore
- Attivazione della CA successiva quando si è certi che i componenti siano stati aggiornati

Non possiamo eseguire queste azioni per tuo conto. I clienti esterni esistono al di fuori dei confini AWS operativi. La configurazione CA trust dei nodi di lavoro non gestiti da EKS Auto Mode o Fargate è impostata al momento dell'avvio o tramite processi di bootstrap controllati solo dall'utente. Ciò è coerente con il funzionamento di TLS Trust: il client detiene il proprio trust store e solo l'amministratore del cliente può aggiornarlo.

Le sezioni che seguono si espandono su ogni lato: cosa AWS fa per te e cosa devi fare insieme a una guida dettagliata su come farlo.

Amazon EKS CA Rotation
AWS manages
EKS cluster control plane
EKS Auto Mode instances
AWS Fargate nodes
CA distribution to managed con
Blocking activation until distrib
Preserving cluster availability th

## What AWS fa per te

AWS gestisce quanto segue durante l'intero ciclo di vita della rotazione della CA per il cluster EKS:

Creazione automatica della CA

Se non aggiungi una CA successore alla tua timeline, ne aggiungeremo una automaticamente quando la CA in uscita del tuo cluster EKS si avvicina alla scadenza. Ciò garantisce che il processo di rotazione inizi con il tempo sufficiente per individuare e aggiornare i clienti prima della scadenza.

Aggiornamenti del piano di controllo

Aggiorniamo automaticamente il piano di controllo del cluster EKS per affidarci alla CA successore. Dopo l'attivazione della CA successore, il piano di controllo emette i certificati firmati dalla CA successore. Non è richiesta alcuna azione da parte dell'utente per il piano di controllo.

Aggiornamenti EKS Auto Mode e Fargate

Aggiorniamo automaticamente i nodi EKS Auto Mode e i pod Fargate per affidarci alla CA successiva. Questi componenti sono completamente gestiti da noi e non richiedono alcuna azione da parte dell'utente durante la rotazione della CA.

Aggiornamenti di EKS Capabilities

Aggiorniamo automaticamente EKS Capabilities (AWS Controllers for Kubernetes (ACK), Argo CD e kro (Kube Resource Orchestrator)) per affidarci alla CA successore. Queste risorse gestite comunicano con il server API del cluster e vengono aggiornate come parte del processo di distribuzione della CA. Non è richiesta alcuna azione da parte dell'utente per le risorse gestite in EKS Capabilities. Per ulteriori informazioni, vedere EKS Capabilities.

Monitoraggio dello stato della distribuzione

Man mano che aggiorniamo i componenti gestiti nel tuo cluster EKS, puoi monitorare l'avanzamento dello stato di distribuzione della CA. Questo ti dice se abbiamo completato la nostra parte della rotazione. La CA successore non può essere attivata finché la distribuzione non è completa.

Built-in garanzie

Abbiamo delle protezioni integrate per proteggere il tuo cluster EKS durante la rotazione:

- La CA successore non può essere attivata finché la distribuzione a tutti i componenti AWS gestiti in EKS non è completa

- Una AWS CA successore aggiunta non può essere eliminata mentre è l'unico successore nel cluster. Questa protezione garantisce che il cluster abbia sempre un percorso CA valido per evitarne la scadenza. Dopo l'attivazione di una CA successore, la CA in uscita può essere eliminata.
- Una CA successore aggiunta dal cliente non può essere eliminata dopo aver raggiunto il limite di due anni prima della scadenza della CA. Dopo questo punto, la CA è protetta dall'eliminazione per garantire che il cluster disponga sempre di una CA successore all'avvicinarsi della scadenza.
- Attiveremo automaticamente la CA successore se la scadenza si avvicina e tu non l'hai attivata tu stesso

Queste misure di sicurezza assicurano che la disponibilità del cluster EKS sia preservata per tutto il ciclo di vita della rotazione, indipendentemente dal fatto che tu agisca o meno.

Notifications

Ti informiamo in ogni fase del ciclo di vita della rotazione di CA. Le notifiche vengono inviate tramite AWS Health, Cluster Insights ed e-mail. Ogni notifica indica cosa è successo, quale azione (se necessaria) è richiesta da parte tua e dove si trova il tuo cluster EKS nella timeline di rotazione.

| Notification | Quando | Significato |
|---|---|---|
| Promemoria di scadenza CA | 2,5 anni prima della scadenza del CA | La CA del cluster EKS ha una data di scadenza definita. Piano di rotazione. |
| CA successore aggiunta | Quando si AWS aggiunge o si aggiunge una CA successor e (aggiunta automatica 2 anni prima della scadenza) | Il processo di rotazione è iniziato. AWS sta distribue ndo la CA successore ai componenti gestiti. |
| Distribuzione completa | Poco dopo l'aggiunta (varia in base al cluster) | AWS ha completato la sua versione. Ora puoi aggiornare i nodi di lavoro che gestisci e i client esterni. |
| Avviso di attivazione | 60 giorni prima dell'attivazione automatica | Attiveremo presto la CA successiva. Aggiorna i |

| Notification | Quando | Significato |
| --- | --- | --- |
| | | componenti se non l'hai già fatto. |
| CA successore attivata | Quando si AWS attiva o si attiva (attivazione automatica 6 mesi prima della scadenza) | Il cluster EKS sta ora emettendo certificati dalla CA successore. |
| Attivazione automatica finale (se ripristinata) | 45 giorni prima della scadenza | AWS attiva la CA successore. Nessun rollback CA disponibile. |

> **Note**
>
> I cluster creati nel 2018-2019 hanno una tempistica di notifica diversa. Questi cluster riceveranno notifiche automatiche in base a una pianificazione modificata.

Puoi anche configurare le tue notifiche utilizzando Amazon EventBridge per integrare gli eventi di rotazione delle CA nei flussi di lavoro di monitoraggio e avviso esistenti.

## Cosa devi fare (e perché)

Una rotazione CA di successo richiede l'aggiornamento dei componenti che AWS non possono essere raggiunti per tuo conto. Questi si dividono in due categorie:

Clienti esterni

Qualsiasi sistema che si connette al server API del cluster EKS dall'esterno del cluster. Ciò include workstation per sviluppatori, CI/CD pipeline (Jenkins, GitHub Actions, ArgoCD) GitLab, strumenti di monitoraggio e osservabilità, script di automazione e qualsiasi applicazione che utilizza un kubeconfig per comunicare con il server API.

Ciascuno di questi sistemi mantiene la propria configurazione di fiducia. Quando la CA successore è attivata, il server API presenta i certificati firmati dalla CA successore. L'aggiornamento di questi client in modo che considerino attendibile la CA successore prima dell'attivazione della CA successore garantisce che mantengano la connettività. Se manca un client, CA rollback può ripristinare l'accesso durante il completamento dell'aggiornamento.

Nodi di lavoro (modalità automatica non EKS, non Fargate)

La configurazione CA trust dei nodi di lavoro non gestiti da EKS Auto Mode o Fargate è impostata al momento dell'avvio o tramite il processo di bootstrap di kubelet. Questi nodi devono essere aggiornati in modo che si fidino della CA successore. L'azione richiesta dipende dal tipo di nodo di lavoro:

Gruppi di nodi gestiti

Esegue un aggiornamento della versione del gruppo di nodi, che attiva una sostituzione progressiva dei nodi. I nuovi nodi vengono avviati automaticamente con la configurazione aggiornata di CA trust.

Karpenter-controlled nodi

Se il rilevamento della deriva è abilitato, Karpenter eseguirà i cicli dei nodi all'interno della finestra di deriva configurata e i nuovi nodi preleveranno la CA successiva senza alcuna azione manuale. Se il rilevamento della deriva è disabilitato o impostato su una finestra lunga, tratta questi nodi come nodi autogestiti.

Self-managed nodi

Sostituisci i nodi in modo che si avviino con la configurazione di fiducia di CA aggiornata. Ciò comporta in genere l'aggiornamento del modello di avvio con i dati CA aggiornati e l'attivazione di una sostituzione progressiva tramite il gruppo Auto Scaling.

Nodi ibridi

Aggiorna la configurazione di trust su ogni nodo ibrido per includere la CA successore. Il processo specifico dipende dal modo in cui i nodi ibridi sono stati avviati e da come viene gestita la loro configurazione di trust.

Puoi identificare quali tipi di nodi di lavoro sono in esecuzione nel tuo cluster EKS utilizzando Cluster Insights. Una guida dettagliata per l'aggiornamento di ciascun tipo è fornita in una sezione successiva.

Forniamo CA rollback come rete di sicurezza in caso di perdita di alcuni nodi di lavoro. Tuttavia, l'aggiornamento di tutti i nodi di lavoro prima dell'attivazione della CA successiva evita completamente le interruzioni della connettività. I nodi non aggiornati prima dell'attivazione della CA successore perderanno la connettività al server API del cluster EKS fino a quando non verranno sostituiti o non verrà eseguito il rollback della CA.

Perché solo tu puoi farlo

I clienti esterni esistono al di fuori dei confini AWS operativi. Una CI/CD pipeline in esecuzione nella rete aziendale, il laptop di uno sviluppatore, uno strumento di monitoraggio ospitato in sede: non AWS dispone di alcun meccanismo per accedere a questi sistemi e aggiornarne la configurazione di fiducia.

Non-EKS La configurazione di attendibilità dei nodi di lavoro in modalità automatica è controllata tramite modelli di avvio, script di dati utente o processi di bootstrap di tua proprietà. L'aggiornamento richiede la sostituzione dei nodi o la modifica della loro configurazione, entrambe azioni all'interno dell'infrastruttura.

Questo è un vincolo del modello di fiducia TLS utilizzato oggi da Kubernetes. Non esiste un meccanismo a livello di protocollo che consenta al server API di verificare se un client ha aggiornato il proprio pacchetto di fiducia. Il server può presentare il proprio certificato solo quando un client si connette. Se il client si fida della CA che lo ha firmato, la connessione ha esito positivo. In caso contrario, fallisce. Non esiste un percorso di verifica pre-attivazione che AWS consenta di confermare la disponibilità dei componenti per conto dell'utente.

Quando iniziare

Inizia a identificare i tuoi clienti esterni il prima possibile. Questa è la parte più dispendiosa in termini di tempo della rotazione delle CA, in particolare in ambienti con più team che distribuiscono in modo indipendente i carichi di lavoro nel cluster EKS. Quanto prima si inizia a individuare i clienti, tanto più tempo si ha a disposizione per coordinare gli aggiornamenti tra i team senza pressioni.

Una guida dettagliata su come aggiornare ogni tipo di client e nodo di lavoro è fornita nelle sezioni che seguono.

## Prerequisiti

Prima di iniziare la rotazione CA, conferma quanto segue:

- AWS CLI:  versione 2.x o successiva. Le API di rotazione CA sono disponibili nella CLI più recente. AWS Esegui `aws --version` per controllare.
- Accesso alla console: la rotazione  CA è disponibile nella console Amazon EKS per le regioni supportate.
- Disponibilità nelle regioni: la rotazione  CA è disponibile in tutte le regioni AWS commerciali in cui Amazon EKS è supportato.

Non sono richieste autorizzazioni IAM aggiuntive oltre a quelle necessarie per gestire il cluster EKS. Se oggi puoi chiamare le API EKS per il tuo cluster EKS, puoi eseguire la rotazione CA.

## Nozioni di base

Puoi eseguire la rotazione della CA utilizzando l'interfaccia a riga di AWS comando o la console Amazon EKS. Assicurati di eseguire la AWS CLI versione 2.x o successiva (`aws --version`per verificare).

Utilizzo di AWS CLI

La seguente procedura dettagliata illustra il processo di rotazione completa della CA utilizzando la CLI e le AWS API EKS.

Passaggio 1: controlla la CA attiva

Visualizza l'autorità di certificazione attiva sul tuo cluster EKS.

```
aws eks list-certificate-authorities --cluster-name my-cluster --region us-west-2
```

Output previsto:

```
{
    "certificateAuthorities": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-example11111",
            "createdAt": "2024-01-15T10:30:00-07:00",
            "createdBy": "EKS",
            "activatedAt": "2024-01-15T10:30:00-07:00",
            "activatedBy": "EKS",
            "signingStatus": "IN_USE",
            "distributionStatus": "COMPLETE"
        }
    ]
}
```

Mostra la CA creata al momento della creazione del cluster EKS. È attualmente in uso (firma dei certificati) e la distribuzione è completa (tutti i componenti AWS gestiti in EKS si fidano di esso).

Fase 2: Visualizza i dettagli e la scadenza della CA

```
aws eks describe-certificate-authority --cluster-name my-cluster --certificate-
authority-id a1b2c3d4-5678-90ab-cdef-example11111 --region us-west-2
```

Output previsto:

```
{
    "certificateAuthority": {
        "id": "a1b2c3d4-5678-90ab-cdef-example11111",
        "createdAt": "2024-01-15T10:30:00-07:00",
        "createdBy": "EKS",
        "activatedAt": "2024-01-15T10:30:00-07:00",
        "activatedBy": "EKS",
        "signingStatus": "IN_USE",
        "distributionStatus": "COMPLETE",
        "validity": {
            "notBefore": "2024-01-15T10:30:00-07:00",
            "notAfter": "2029-01-14T10:30:00-07:00"
        },
        "rollbackAvailable": false
    }
}
```

Il `validity` blocco mostra quando è stata creata la CA (`notBefore`) e quando scade (`notAfter`). `rollbackAvailable`indica se è possibile ripristinare una CA precedente dopo l'attivazione della CA successiva. Per la CA iniziale creata con il cluster EKS, ciò sarà `false` dovuto al fatto che non esiste una CA precedente a cui tornare.

> **Note**
>
> Il `scheduledEvents` blocco (contenente `firstAutoActivation` `efinalAutoActivation`) appare sulla CA successore, non sulla CA in uscita. Questi campi mostrano quando attiveremo automaticamente la CA successore se non l'hai fatto tu stesso. Vedrai questi campi quando descrivi la CA successore dopo averla aggiunta.

Fase 3: aggiungere una CA successore

```
aws eks create-certificate-authority --cluster-name my-cluster --region us-west-2
```

Questo aggiunge una CA successore al cluster EKS. La risposta include una `updateId` che puoi usare per tenere traccia dei progressi.

Fase 4: Tieni traccia dell'aggiornamento

```
aws eks describe-update --name my-cluster --update-id a1b2c3d4-update-id --region us-west-2
```

Attendi che lo stato dell'aggiornamento sia`Successful`.

> **Note**
>
> Se viene visualizzato lo stato dell'aggiornamento `UPDATE_FAILED` e viene visualizzata la CA successore`FAILED`, la creazione della CA non è riuscita. `distributionStatus` Elimina la CA fallita utilizzando `aws eks delete-certificate-authority` e creane una nuova. Per le rotazioni automatiche AWS avviate, rileva e pulisce AWS automaticamente le CA guaste prima di aggiungere un nuovo successore, quindi in questo scenario non è richiesta alcuna azione da parte del cliente.

Fase 5: Verifica lo stato della distribuzione

```
aws eks list-certificate-authorities --cluster-name my-cluster --region us-west-2
```

Ora dovresti vedere due CA. La CA successore avrà `signingStatus: NOT_USED` e `distributionStatus` procederà `COMPLETE` successivamente `IN_PROGRESS` AWS all'aggiornamento di tutti i componenti gestiti nel cluster EKS.

Non procedere finché non sarà raggiunto lo stato di distribuzione della CA successore. `COMPLETE`

Passaggio 6: aggiorna il tuo kubeconfig

```
aws eks update-kubeconfig --name my-cluster --region us-west-2
```

Questo aggiorna il tuo kubeconfig locale in modo che entrambe le CA siano affidabili. Dopodiché, i comandi kubectl continueranno a funzionare dopo l'attivazione della CA successiva.

Passaggio 7: aggiorna i nodi di lavoro che gestisci e i client esterni

Questo è trattato in dettaglio nella sezione successiva. Dopo che tutti i nodi di lavoro gestiti e i client esterni sono stati aggiornati in modo da considerare attendibili la CA successore, procedi all'attivazione.

Fase 8: Attivazione della CA successore

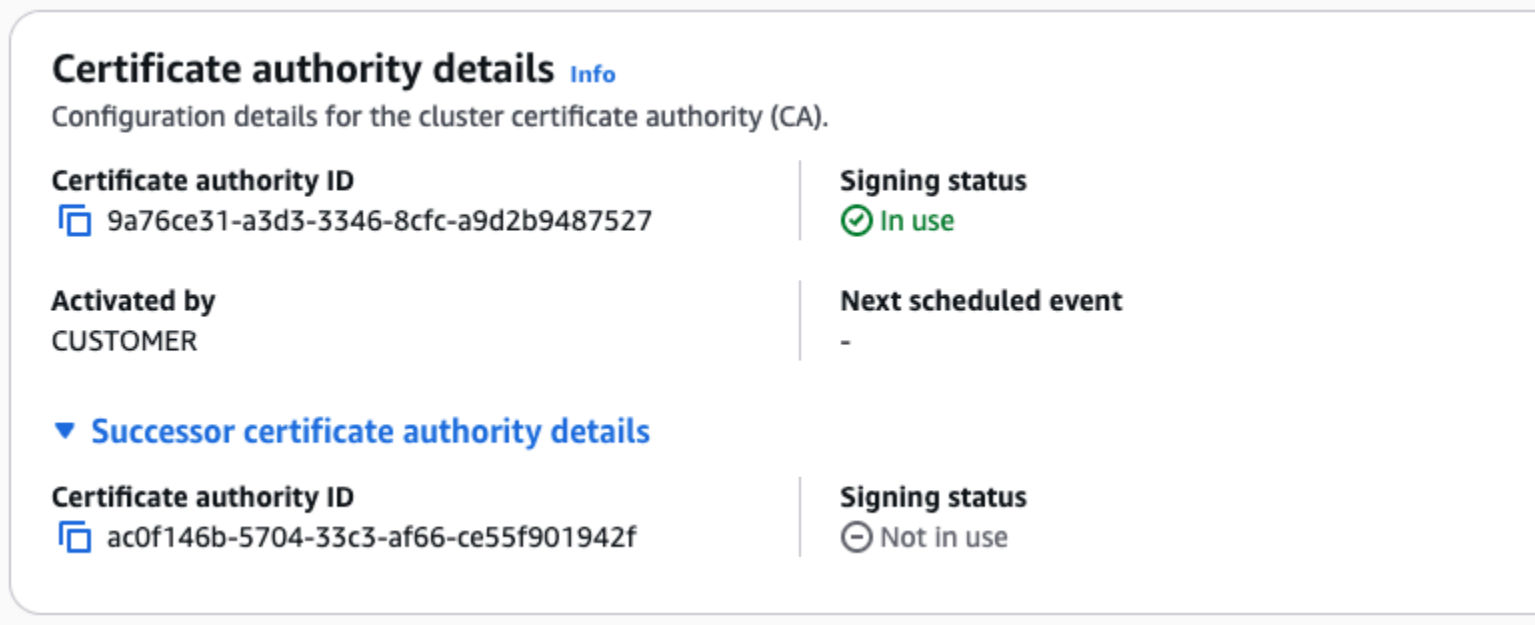
```
aws eks activate-certificate-authority --cluster-name my-cluster --certificate-
authority-id <successor-ca-id> --region us-west-2
```

Dopo l'attivazione della CA successore, il cluster EKS emette i certificati firmati dalla CA successore. Verifica la connettività per confermare che tutti i componenti funzionino come previsto.

Utilizzo della console Amazon EKS

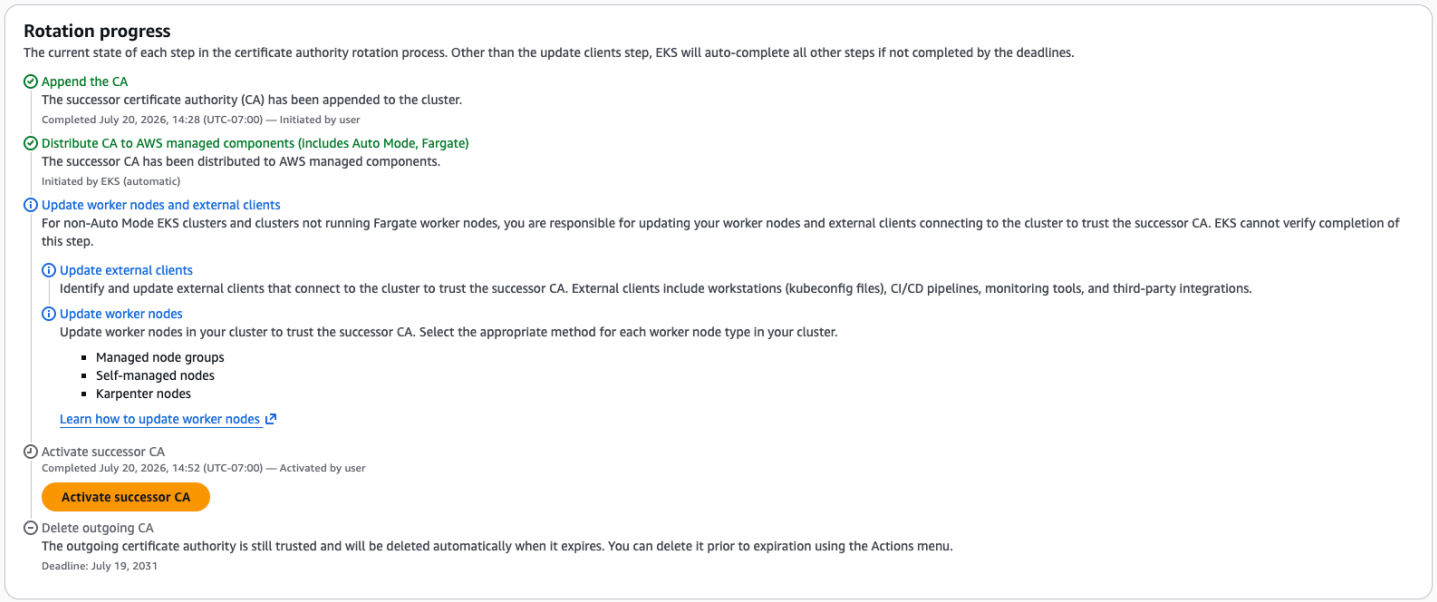
La console Amazon EKS offre un'esperienza guidata per la rotazione delle CA. Puoi visualizzare lo stato della CA, aggiungere una CA successore, monitorare l'avanzamento della distribuzione e attivare la CA successore direttamente dalla console.

L'immagine seguente mostra la visualizzazione dei dettagli dell'autorità di certificazione nella console di Amazon EKS durante una rotazione attiva. La CA attiva e la CA successore vengono entrambe visualizzate con lo stato di firma, la data di scadenza e i giorni mancanti alla scadenza.




L'immagine seguente mostra la visualizzazione dell'avanzamento della rotazione nella console Amazon EKS. Ogni fase del processo di rotazione viene visualizzata con il suo stato attuale, inclusi

l'aggiunta, la distribuzione, l'aggiornamento dei nodi di lavoro e dei client esterni, l'attivazione e l'eliminazione della CA in uscita.




## Aggiornamento dei client Kubernetes

> **Important**
>
> Durante il periodo di doppia fiducia, il pacchetto di fiducia del cluster contiene due certificati CA. La dimensione combinata di due CA con codifica base64 è di circa 2,8 KB (o circa 1,9 KB con compressione gzip). Per i nodi di lavoro in cui vengono forniti dati utente personalizzati nei modelli di avvio EC2, verifica che la dimensione totale dei dati utente non superi il limite dei dati utente EC2 di 16 KB. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html Se i dati utente esistenti sono vicini a questo limite, valuta la possibilità di comprimere il contenuto dei dati utente utilizzando gzip per ridurne le dimensioni.

Dopo aver aggiunto la CA successore e aver AWS completato la distribuzione ai componenti gestiti nel cluster EKS (`distributionStatus: COMPLETE`), è necessario aggiornare i propri componenti per rendere attendibile la CA successore. Un «client» in questo contesto è qualsiasi sistema che si connette al server API del cluster EKS. Ciò include la configurazione locale di kubectl, le CI/CD pipeline, gli strumenti di monitoraggio, gli script di automazione e i nodi di lavoro.

I dati CA aggiornati per il cluster EKS (che ora contiene sia la CA attuale che quella successiva) possono essere recuperati utilizzando:

```
aws eks describe-cluster --name my-cluster --region us-west-2 --query
 'cluster.certificateAuthority.data' --output text
```

Utilizzate questo valore per aggiornare la configurazione di trust di ogni tipo di client nelle seguenti sottosezioni.

Kubeconfig (workstation per sviluppatori, pipeline, automazione) CI/CD

Esegui quanto segue per aggiornare il tuo kubeconfig locale:

```
aws eks update-kubeconfig --name my-cluster --region us-west-2
```

Questo recupera automaticamente i dati CA più recenti e aggiorna il tuo kubeconfig. Qualsiasi sistema che utilizza questo kubeconfig si fiderà di entrambe le CA.

Per CI/CD le pipeline e le automazioni che generano il proprio kubeconfig (ad esempio, utilizzando direttamente le API EKS o archiviando kubeconfig come segreto), aggiorna il campo con il valore recuperato da. `certificate-authority-data` `describe-cluster`

Gruppi di nodi gestiti

Esegui un aggiornamento della versione del gruppo di nodi per attivare una sostituzione progressiva dei nodi:

```
aws eks update-nodegroup-version --cluster-name my-cluster --nodegroup-name my-
nodegroup --region us-west-2
```

I nuovi nodi si avviano automaticamente con i dati CA aggiornati. La sostituzione progressiva garantisce che i nodi vengano sostituiti uno alla volta senza interrompere i carichi di lavoro in esecuzione.

Se gestisci i tuoi gruppi di nodi tramite Terraform o CloudFormation, la rotazione CA non crea deviazioni nello stato IaC. Il ciclo di vita della CA è gestito tramite API EKS dedicate, separate dalla configurazione delle risorse del cluster. Per maggiori dettagli su come la rotazione di CA interagisce con l'infrastruttura come codice, consulta la sezione Infrastructure as code.

Modello di avvio personalizzato con un'AMI personalizzata

Se il gruppo di nodi è distribuito con un'AMI personalizzata, AWS non unisce i dati utente. L'utente è responsabile della fornitura della corretta configurazione di bootstrap, incluso il CA trust bundle

aggiornato. CA rotation non aggiorna automaticamente i dati utente e un nodo senza la CA successore non riesce a entrare nel cluster.

1. Recupera i dati CA aggiornati (il trust bundle combinato contenente le CA in uscita e quelle successive):

```
aws eks describe-cluster --name my-cluster --region us-west-2 --query
 'cluster.certificateAuthority.data' --output text
```

2. Aggiorna i dati CA nei dati utente del modello di avvio. Il modo in cui specifichi i dati CA dipende dal sistema operativo e dal meccanismo di bootstrap e corrisponde a come li hai forniti originariamente. Per ulteriori informazioni sulla personalizzazione dei nodi gestiti, consulta Personalizzare i nodi gestiti con modelli di avvio.
3. Crea una nuova versione del modello di avvio con i dati utente aggiornati, quindi aggiorna il gruppo di nodi a quella versione del modello di avvio. Questo ricicla i nodi in modo che vengano avviati con la CA successiva:

```
aws eks update-nodegroup-version --cluster-name my-cluster --nodegroup-name my-
nodegroup --launch-template id=<lt-id>,version=<new-version> --region us-west-2
```

   Per ulteriori informazioni sull'aggiornamento di un gruppo di nodi a una nuova versione del modello di avvio, consulta Aggiornare un gruppo di nodi gestito per il cluster.

Verifica che i nodi siano in esecuzione sul modello di avvio aggiornato

Prima di procedere all'attivazione, verificate che tutti i nodi del gruppo di nodi gestiti siano in esecuzione sulla versione più recente del modello di avvio. Questa versione deve contenere il CA trust bundle aggiornato.

1. Ottieni il modello di avvio per il gruppo di nodi. Se `describe-nodegroup` restituisce un `launchTemplate` campo, usalo direttamente:

```
aws eks describe-nodegroup --cluster-name my-cluster --nodegroup-name my-nodegroup --
region us-west-2 --query 'nodegroup.launchTemplate'
```

   Se non restituisce un `launchTemplate` campo, AWS gestisce il modello di avvio internamente. Trovalo invece tramite il gruppo Auto Scaling:

```
ASG=$(aws eks describe-nodegroup --cluster-name my-cluster --nodegroup-name my-
nodegroup --region us-west-2 --query 'nodegroup.resources.autoScalingGroups[0].name'
 --output text)
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-
names "$ASG" --region us-west-2 --query 'AutoScalingGroups[0].
{LaunchTemplate: LaunchTemplate, MixedInstancesPolicy:
 MixedInstancesPolicy.LaunchTemplate.LaunchTemplateSpecification}'
```

> **Note**
>
> Il modello di avvio potrebbe rientrare `LaunchTemplate` o `MixedInstancesPolicy` dipendere dalla configurazione del gruppo Auto Scaling.

2. Decodifica i dati utente del modello di avvio. Usa l'ID e la versione del modello di avvio indicati nel passaggio precedente. Verifica che i dati CA nei dati utente corrispondano al pacchetto di fiducia combinato restituito da`describe-cluster`. Il campo che contiene i dati CA dipende dal sistema operativo e dal meccanismo di bootstrap in uso:

```
aws ec2 describe-launch-template-versions --launch-template-
id <lt-id> --versions <version> --region us-west-2 --query
 'LaunchTemplateVersions[0].LaunchTemplateData.UserData' --output text | base64 --
decode
```

3. Verifica che tutti i nodi di lavoro siano in esecuzione sulla versione più recente del modello di avvio dopo l'aggiornamento. Descrivi il gruppo Auto Scaling per il gruppo di nodi. Confronta la versione del modello di avvio di ciascuna istanza con la versione corrente del modello di avvio del gruppo. Ogni `InService` istanza deve essere nella versione corrente. La sostituzione a rotazione scarica le istanze in uno `Terminating` stato. Puoi ignorare queste istanze:

```
ASG=$(aws eks describe-nodegroup --cluster-name my-cluster --nodegroup-name my-
nodegroup --region us-west-2 --query 'nodegroup.resources.autoScalingGroups[0].name'
 --output text)
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names "$ASG" --
region us-west-2 --query 'AutoScalingGroups[0].Instances[].{InstanceId: InstanceId,
 LifecycleState: LifecycleState, LaunchTemplateVersion: LaunchTemplate.Version}' --
output table
```

4. Verificate che i nodi sostitutivi siano integri. Ogni nodo del gruppo di nodi deve esserloReady, il che conferma che il kubelet ha stabilito una connessione al server API utilizzando il CA trust bundle aggiornato:

```
kubectl get nodes -l eks.amazonaws.com/nodegroup=my-nodegroup
```

Karpenter-controlled nodi

Se il rilevamento della deriva è abilitato in Karpenter NodePool, Karpenter rileverà automaticamente che i nodi sono in esecuzione con dati CA obsoleti e li ciclerà entro la finestra di interruzione configurata. I nuovi nodi raccoglieranno la CA successiva senza alcuna azione manuale.

Verifica che il rilevamento della deriva sia abilitato nella tua NodePool configurazione:

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: default
spec:
  disruption:
    consolidationPolicy: WhenEmptyOrUnderutilized
    budgets:
      - nodes: "10%"
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: default
```

Se `disruption` è configurato con una politica di consolidamento, il rilevamento della deriva è attivo per impostazione predefinita. Karpenter sostituirà i nodi che sono passati dallo stato desiderato, che include le modifiche ai dati CA.

Se il rilevamento della deriva è abilitato, verifica che il budget previsto per le interruzioni consenta la sostituzione di tutti i Karpenter-controlled nodi prima della data di attivazione della CA successiva. Se il budget è troppo restrittivo (ad esempio, una finestra di manutenzione ristretta con una percentuale di sostituzione bassa), non tutti i nodi potrebbero essere sostituiti in tempo.

Se il rilevamento della deriva è disabilitato o i budget previsti per le interruzioni limitano le sostituzioni a una finestra che supera la tempistica di rotazione prevista, puoi attivare manualmente la sostituzione dei nodi isolando e svuotando i nodi:

```
kubectl cordon <node-name>
kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-data
```

Karpenter fornirà un nodo sostitutivo che si avvia con i dati CA aggiornati.

Self-managed nodi

Per i nodi autogestiti, è necessario aggiornare i dati della CA nel modello di avvio o nello script dei dati utente utilizzati dai nodi durante il bootstrap:

1. Recupera i dati CA aggiornati:

   ```
   aws eks describe-cluster --name my-cluster --region us-west-2 --query
    'cluster.certificateAuthority.data' --output text
   ```

2. Aggiorna il modello di avvio (o i dati utente) con il valore dei dati CA aggiornato.
3. Attiva una sostituzione progressiva dei nodi tramite il tuo gruppo Auto Scaling (ad esempio, aggiornamento dell'istanza).

I nuovi nodi verranno avviati con i dati CA aggiornati e si affideranno sia alle CA attuali che a quelle successive.

AWS Pod Fargate (tipo di lancio EKS Fargate)

AWS I pod Fargate del cluster EKS funzionano in modo diverso dalle altre modalità di avvio del piano dati EKS (gruppi di nodi gestiti, nodi autogestiti, nodi). Karpenter-controlled

Quando un pod si connette al server API, accadono due cose: il pod si autentica utilizzando il token dell'account di servizio (dimostrando la propria identità al server API) e il pod verifica l'identità del server API controllando che il certificato del server sia stato firmato da una CA di cui si fida. I dati CA archiviati nell'ambiente del pod sono ciò che rende possibile questa verifica. Se il server API inizia a presentare certificati firmati da una CA successore di cui il pod non considera attendibili, il pod rifiuterà la connessione.

In altre modalità di avvio del piano dati EKS (gruppi di nodi gestiti, nodi autogestiti, Karpenter-controlled nodi), i dati CA risiedono sul nodo. Quando si sostituisce il nodo, il nuovo nodo si avvia con

i dati CA aggiornati. I pod pianificati su quel nuovo nodo ricevono i dati CA aggiornati, consentendo loro di verificare il server API indipendentemente dalla CA che ha firmato il certificato.

In Fargate, ogni pod viene eseguito nel proprio ambiente di calcolo dedicato con il proprio processo kubelet. Questo kubelet si avvia con i dati della CA al momento della creazione del pod. Al di sotto non è presente alcun nodo condiviso e non si ha accesso diretto al calcolo sottostante.

Non è richiesta alcuna azione da parte del cliente per i nodi AWS Fargate in EKS durante la rotazione CA.  AWS ricicla naturalmente i pod nei nodi Fargate attraverso il processo di patching. Dopo l'aggiunta di una CA successiva, i pod Fargate preesistenti vengono riciclati come parte di questo processo. Si affideranno alla CA successore senza alcuna azione da parte tua. Poiché AWS aggiunge la CA successore con largo anticipo rispetto a qualsiasi tempistica di attivazione della CA AWS avviata in EKS, i pod Fargate saranno stati riciclati e si affideranno alla CA successore nel momento in cui attiverà la CA successore. AWS È presente una protezione integrata che impedisce l'attivazione della CA successiva fino al termine del riciclaggio dei pod Fargate.

Ciò significa che entrambe le opzioni del piano dati gestito (EKS Auto Mode e Fargate) per un cluster EKS offrono la stessa esperienza utente per la rotazione della CA: non è richiesta alcuna azione da parte del cliente per i nodi di lavoro. Sei comunque responsabile dell'aggiornamento di tutti i client esterni che si connettono al server API.

Questo è un caso limite. Data la tempistica di rotazione (la CA successore è stata aggiunta anni prima della scadenza), il naturale ciclo di patching si concluderà ben prima dell'attivazione della CA successiva nella maggior parte degli scenari. La protezione esiste come misura preventiva nell'improbabile caso in cui un cliente tenti l'attivazione anticipata.

Client esterni (strumenti di monitoraggio, integrazioni di terze parti)

Qualsiasi applicazione o strumento che si connette al server API del cluster EKS utilizzando una configurazione kubeconfig o certificate trust deve essere aggiornato con i dati CA aggiornati. Questo include:

- Strumenti di monitoraggio e osservabilità (agenti Datadog, Prometheus, Grafana)
- GitOps controller che funzionano all'esterno del cluster (ArgoCD, Flux)
- Automazione o script personalizzati che richiamano l'API Kubernetes
- Qualsiasi sistema che viene `certificate-authority-data` memorizzato come valore statico

Per ognuno di questi, sostituisci i dati CA archiviati con il valore aggiornato di`describe-cluster`.

Come verificare che un client sia stato aggiornato

Dopo aver aggiornato un client, verifica che possa ancora comunicare con il server API:

```
kubectl get nodes
```

Se il comando ha esito positivo, il tuo kubeconfig si fida dei dati CA attivi. Dopo l'attivazione della CA successiva, esegui lo stesso comando per confermare la continuità della connettività.

## Infrastructure as code (IaC)

È possibile eseguire la rotazione della CA insieme alla propria infrastruttura di codice (IaC) esistente senza creare deviazioni o richiedere modifiche alle configurazioni IaC.

Perché la rotazione CA non influisce sullo stato IaC

Il ciclo di vita della CA è gestito tramite API EKS dedicate (`create-certificate-authority`,`activate-certificate-authority`,`delete-certificate-authority`) completamente separate dalla configurazione delle risorse del cluster EKS. Indipendentemente dal fatto che la rotazione della CA sia avviata dall'utente o automaticamente da AWS, nessuna proprietà tracciata dagli strumenti IaC sulla risorsa del cluster EKS viene modificata.

Ciò significa che:

- L'applicazione o l'aggiornamento dello stack IaC dopo l'aggiunta o l'attivazione di una CA non rileverà una deriva o un tentativo di riconciliazione dello stato della CA
- Le operazioni di rotazione della CA avviate tramite CLI o Console non sono in conflitto con le risorse del cluster IaC-managed

Il `certificateAuthority.data` campo restituito da `describe-cluster` è un output di sola lettura. Riflette l'attuale pacchetto di fiducia combinato (entrambe le CA durante il periodo di doppia fiducia) ma non è una proprietà configurabile. Gli strumenti IaC non lo considerano qualcosa da riconciliare.

I campi di attribuzione (`createdBy`,`activatedBy`) in ogni record CA consentono di distinguere tra le operazioni avviate dall'utente e le operazioni AWS avviate automaticamente, il che supporta i flussi di lavoro di audit e gestione delle modifiche.

### Utilizzo con rotazione CA CloudFormation

La rotazione della CA può essere attivata CloudFormation utilizzando una WriteOnly proprietà sulla `AWS::EKS::Cluster` risorsa. Questa proprietà attiva l'attivazione ma non viene memorizzata nello stato dello stack, pertanto gli aggiornamenti successivi dello stack senza di essa non tentano di ripristinarla o disattivarla.

```
# Phase 1: Add to existing stack that manages your cluster
Resources:
    NewCA:
        Type: AWS::EKS::CertificateAuthority
        Properties:
            ClusterName: my-cluster

    MyCluster:
        Type: AWS::EKS::Cluster
        Properties:
            Name: my-cluster
```

Questo primo aggiornamento dello stack aggiunge una CA successore al cluster. Attendi il raggiungimento `COMPLETE` dello stato di distribuzione della CA successore prima di procedere.

```
# Phase 2: After distribution completes, update your existing Cluster resource to
 activate
Resources:
    NewCA:
        Type: AWS::EKS::CertificateAuthority
        Properties:
            ClusterName: my-cluster

    # Update your existing Cluster resource to activate
    MyCluster:
        Type: AWS::EKS::Cluster
        Properties:
            Name: my-cluster
            ActiveCertificateAuthorityId: !GetAtt NewCA.Id
```

Questo secondo aggiornamento dello stack attiva l'attivazione della CA successore. Poiché `ActiveCertificateAuthorityId` è una WriteOnly proprietà, non viene restituita in lettura e non CloudFormation rileverà la deriva se la CA attiva cambia all'esterno CloudFormation (ad esempio, tramite l'attivazione automatica da parte di). AWS

Importante: l' CloudFormation integrazione della rotazione della CA segue uno schema diverso rispetto alle risorse tipiche CloudFormation . Un'autorità di certificazione in EKS non è una risorsa indipendente con un proprio ARN. Esiste come parte del ciclo di vita del certificato del cluster ed è autorizzata tramite il cluster stesso, in modo simile a come una policy di ruolo IAM è autorizzata tramite il ruolo principale (`AWS::IAM::RolePolicy`) o un'associazione EIP è autorizzata tramite la sua istanza (). `AWS::EC2::EIPAssociation` I clienti che gestiscono la rotazione tramite CA CloudFormation devono essere consapevoli di questa distinzione.

Gruppi di nodi gestiti e IaC

L'esecuzione di un aggiornamento della versione del gruppo di nodi per aggiornare i nodi con la configurazione CA trust aggiornata è un'azione operativa. I nuovi nodi si avviano automaticamente con i dati di trust CA attivi dal cluster. Se i modelli IaC non codificano i dati CA nei modelli di avvio o nei dati utente, non sono necessarie modifiche al modello.

## Rollback CA

Dopo aver attivato una CA successore, puoi eseguire il rollback alla CA precedente se rilevi problemi di connettività con i nodi di lavoro che gestisci (modalità automatica non EKS, non Fargate) o client esterni. Il rollback riattiva la CA precedente come autorità di firma per il cluster EKS.

Quando il rollback di CA è disponibile

Il rollback CA è disponibile dopo l'attivazione di CA a condizione che:

- L'attivazione della CA è stata avviata dal cliente o è stata la prima attivazione automatica entro AWS (circa 6 mesi prima della scadenza della CA in uscita)
- La finestra di rollback non è scaduta

Puoi verificare se il rollback di CA è disponibile in qualsiasi momento utilizzando il `rollbackAvailable` campo restituito da: `describe-certificate-authority`

```
aws eks describe-certificate-authority --cluster-name my-cluster --certificate-
authority-id <ca-id> --region us-west-2
```

```
{
    "certificateAuthority": {
        "id": "a1b2c3d4-5678-90ab-cdef-example22222",
        "signingStatus": "IN_USE",
```

```
        "distributionStatus": "COMPLETE",
        "rollbackAvailable": true
    }
}
```

Quando il rollback di CA non è disponibile

Il rollback di CA non è disponibile dopo l'attivazione automatica finale. Se AWS attiva la CA successore per l'ultima volta (45 giorni prima della scadenza della CA), la rotazione deve procedere in avanti. L'attivazione automatica finale si verifica solo se la prima attivazione automatica è stata precedentemente annullata. Esiste come ultima risorsa di protezione per garantire che il cluster EKS non raggiunga la scadenza della CA senza una CA valida.

Come eseguire il rollback

Per eseguire il rollback, riattiva la CA precedente:

```
aws eks activate-certificate-authority --cluster-name my-cluster --certificate-
authority-id <previous-ca-id> --region us-west-2
```

Cosa succede durante il rollback della CA

- La CA precedente riprende a firmare i certificati per il cluster EKS
- La CA successore rimane nel trust bundle (entrambe le CA sono ancora affidabili)
- I nodi di lavoro e i client che sono già stati aggiornati in modo da considerare attendibili la CA successiva continueranno a funzionare (si affidano a entrambe le CA)
- I nodi di lavoro e i client che non erano ancora stati aggiornati riprenderanno il normale funzionamento (il server API presenta i certificati firmati dalla CA di cui si fidano già)
- I processi Kubelet sui nodi di lavoro si riconnetteranno automaticamente tramite il ciclo di riprova integrato

Quando prendere in considerazione il rollback di CA

Il rollback di CA è un meccanismo di sicurezza per le situazioni in cui l'attivazione della CA successiva rivela un problema di connettività che non hai rilevato in precedenza:

- Un client esterno che non è stato identificato durante la fase di aggiornamento perde la connettività dopo l'attivazione della CA successiva

- Uno strumento di monitoraggio o osservabilità non riesce a convalidare il nuovo certificato
- Una CI/CD pipeline si interrompe perché utilizza una configurazione di attendibilità dei certificati codificata

Dopo il rollback, viene mantenuto l'intero periodo di dual trust per identificare e risolvere il problema prima di riattivare la CA successiva.

## Ripristino senza rollback della CA

Se si attiva la CA successiva prima di aggiornare i gruppi di nodi gestiti e la finestra di rollback della CA non è più disponibile (ad esempio, dopo la scadenza finale per l'attivazione automatica), è possibile eseguire il ripristino eseguendo un aggiornamento continuo sui gruppi di nodi interessati. Tuttavia, il tentativo iniziale di aggiornamento progressivo fallirà perché i nodi disconnessi non possono ricevere comandi di rimozione dei pod dal server API.

Fasi di ripristino

1. Identifica i nodi che sono NotReady:

```
kubectl get nodes
```

2. Elenca i pod su ogni NotReady nodo:

```
kubectl get pods --all-namespaces --field-selector spec.nodeName=<node-name>
```

3. Forza l'eliminazione di tutti i pod su ogni NotReady nodo:

```
kubectl delete pod --force --grace-period=0 -n <namespace> <pod-name>
```

4. Riprova l'aggiornamento progressivo:

```
aws eks update-nodegroup-version --cluster-name <cluster> --nodegroup-name
 <nodegroup> --region <region>
```

5. Verifica il ripristino dei nodi:

```
kubectl get nodes
```

> **Important**
>
> L'eliminazione forzata dei pod interrompe i carichi di lavoro in modo non corretto. La perdita di dati è possibile per carichi di lavoro statici. I contenitori potrebbero continuare a funzionare sull'istanza disconnessa fino a quando non viene terminata dal gruppo Auto Scaling. Questa è l'ultima risorsa quando la finestra di rollback di CA non è più disponibile.

## Considerazioni e limitazioni

### Disponibilità regionale

La rotazione CA è disponibile in tutte le regioni AWS commerciali in cui Amazon EKS è supportato.

### Massimo due CA

Un cluster EKS può avere al massimo due CA in qualsiasi momento: la CA attiva e una successore. Non è possibile aggiungere una seconda CA successore fino al completamento della rotazione precedente.

### Nessuna revoca del certificato

La rotazione CA non supporta la revoca di singoli certificati. Ciò è coerente con Kubernetes a monte, che non implementa la revoca dei certificati (CRL o OCSP). La rotazione sostituisce l'intera CA, che naturalmente invalida tutti i certificati firmati dalla CA in uscita dopo che è stata rimossa dal trust bundle.

### AWS-le CA aggiunte non possono essere eliminate

Se AWS viene aggiunta automaticamente una CA successore, non è possibile eliminarla. Questa protezione garantisce che il processo di rotazione non possa essere interrotto da un'eliminazione accidentale. Customer-appended Le CA possono essere eliminate purché non siano la CA di firma attiva.

### Periodo di validità della CA

La CA originale creata con il cluster ha un periodo di validità di 10 anni. Le CA successive create tramite il processo di rotazione hanno periodi di validità di 5 anni. Tutte le CA future per il cluster seguiranno il periodo di validità di 5 anni. Verifica la scadenza della CA utilizzando. `describe-certificate-authority`

Dimensione dei dati utente EC2 durante il dual trust

Durante il periodo di doppia fiducia, il trust bundle del cluster aumenta di dimensioni a causa della presenza di due certificati CA (circa 2,8 KB combinati o circa 1,9 KB con compressione gzip). Per i nodi di lavoro in cui vengono forniti dati utente personalizzati nei modelli di avvio EC2, verifica che la dimensione totale dei dati utente non superi il limite dei dati utente EC2 di 16 KB. Se i dati utente esistenti sono vicini a questo limite, l'aggiunta di una seconda CA potrebbe causare il fallimento della creazione del modello di avvio e impedire il provisioning di nuovi nodi. Valuta la possibilità di comprimere il contenuto dei dati utente utilizzando gzip per ridurne le dimensioni.

Aggiornamenti della versione durante la rotazione della CA

Gli aggiornamenti della versione del cluster EKS e la rotazione della CA sono operazioni indipendenti. Tuttavia, non è possibile eseguire entrambe le operazioni contemporaneamente. Se è in corso un'operazione di rotazione della CA, l'aggiornamento della versione verrà rifiutato fino al completamento dell'operazione CA e viceversa.

Porta la tua CA (BYOCA)

L'utilizzo della propria CA AWS privata per il backup dei certificati del cluster EKS non è attualmente supportato.

## Domande frequenti

Come posso iniziare con la rotazione di CA?

Per iniziare, corri `aws eks list-certificate-authorities --cluster-name my-cluster` per visualizzare la CA attiva e la sua scadenza. Se sei pronto per iniziare la rotazione, corri `aws eks create-certificate-authority --cluster-name my-cluster` ad aggiungere una CA successore. La procedura dettagliata completa è descritta nella sezione Guida introduttiva. Puoi anche eseguire la rotazione della CA tramite la console Amazon EKS.

Sono previsti costi associati alla rotazione della CA?

No La rotazione CA è disponibile senza costi aggiuntivi per tutti i cluster EKS.

Cosa succede se non faccio ruotare la mia CA prima che scada?

Disponiamo di protezioni automatiche che impediscono al cluster EKS di raggiungere la scadenza della CA senza una CA valida. Se non avvii tu stesso la rotazione delle CA, aggiungeremo e

attiveremo automaticamente una CA successore prima della scadenza, assicurando che il cluster rimanga disponibile.

Tuttavia, una rotazione corretta richiede anche l'aggiornamento dei nodi di lavoro gestiti (modalità automatica non EKS, non Fargate) e i client esterni affinché si fidino della CA successore. Se questi componenti non vengono aggiornati prima dell'attivazione della CA successore, perderanno la connettività al server API.

In Kubernetes, se una CA non viene ruotata prima della scadenza, tutti i certificati firmati da tale CA diventano non validi. Il server API non può più essere raggiunto da nessun client e il cluster diventa non disponibile.

Quanto tempo ho a disposizione per completare la rotazione della CA?

Il tempo complessivo necessario per completare la rotazione della CA dipende sia dalle misure di protezione AWS automatizzate che dal processo di aggiornamento personalizzato.

AWS fornisce tempistiche definitive per ciò che gestisce. Una CA successore viene aggiunta circa 2 anni prima della scadenza della CA in uscita. Se non attivi tu stesso la CA successore, la attiveremo automaticamente circa 6 mesi prima della scadenza. Se effettui il rollback dopo l'attivazione automatica, eseguiremo un'attivazione automatica definitiva 45 giorni prima della scadenza. Queste misure di sicurezza assicurano che il cluster EKS rimanga disponibile indipendentemente dal fatto che tu agisca.

Una corretta rotazione delle CA dipende anche dall'aggiornamento dei nodi di lavoro gestiti (modalità automatica non EKS, non Fargate) e dai client esterni affinché si fidino della CA successore prima dell'attivazione della CA successore. Il tempo necessario dipende dalla configurazione del nodo di lavoro del piano dati, dall'impronta del client esterno e dal tempo necessario per eseguire l'individuazione e gli aggiornamenti di questi componenti.

La rotazione della CA causa tempi di inattività del mio cluster?

No. Il cluster EKS rimane disponibile per l'intero ciclo di vita di rotazione di CA. Il piano di controllo continua a soddisfare le richieste in ogni fase. Durante il periodo di doppia fiducia, sia le CA in uscita che quelle successive vengono considerate attendibili contemporaneamente, consentendo l'aggiornamento incrementale dei componenti senza interrompere le operazioni del cluster. Se si esegue il rollback alla CA precedente a causa di problemi sul lato client, AWS esegue un rollforward finale circa 45 giorni prima della scadenza della CA in uscita a titolo di salvaguardia.

È importante distinguere tra il cluster EKS (piano di controllo) e i componenti del piano dati. Il piano di controllo è completamente gestito da AWS e rimane disponibile per tutta la rotazione.

Per EKS Auto Mode e Fargate, AWS aggiorna automaticamente i nodi di lavoro. Non vi è alcun rischio di perdita di connettività per i nodi di lavoro in queste modalità di avvio del piano dati. Sei comunque responsabile dell'aggiornamento di tutti i client esterni che si connettono al server API.

Per i gruppi di nodi gestiti, i nodi autogestiti, Karpenter-controlled le istanze (senza il rilevamento della deriva abilitato) e i nodi ibridi, sei responsabile della sostituzione o dell'aggiornamento di tali nodi prima dell'attivazione della CA successore. Se non vengono sostituiti, tali nodi perderanno la connettività al piano di controllo dopo l'attivazione della CA successiva, anche se il piano di controllo stesso rimane completamente operativo.

I miei carichi di lavoro verranno interrotti durante la rotazione della CA?

I carichi di lavoro in esecuzione (pod) non vengono interrotti dalla rotazione di CA stessa. I pod comunicano tra loro attraverso la rete del cluster, che non è interessata da una modifica della CA. La CA viene utilizzata per la comunicazione tra i componenti e il server API, non per il traffico pod-to-pod.

Se i nodi di lavoro devono essere sostituiti come parte dell'aggiornamento per rendere attendibili la CA successore (ad esempio, gruppi di nodi gestiti che eseguono un aggiornamento progressivo o Karpenter che sostituisce i nodi deviati), i pod su tali nodi verranno riprogrammati come parte del normale processo di sostituzione dei nodi. Questo è il comportamento standard di Kubernetes durante la sostituzione dei nodi, non un effetto collaterale della rotazione della CA. Assicurati di avere i Pod Disruption Budgets (PDB) configurati per i carichi di lavoro critici per controllare in che modo i pod vengono eliminati durante la sostituzione dei nodi.

Quando scadrà la CA del mio cluster?

Puoi verificare quando scade la CA del tuo cluster utilizzando la AWS CLI, le API EKS o la console Amazon EKS. Ad esempio, puoi eseguire quanto segue:

```
aws eks describe-certificate-authority --cluster-name my-cluster
 --certificate-authority-id <ca-id> --region us-west-2 --query
 'certificateAuthority.validity.notAfter'
```

Puoi trovare l'ID della tua CA eseguendo`aws eks list-certificate-authorities --cluster-name my-cluster`.

Qual è il periodo di validità della CA del mio cluster?

La CA originale creata con il cluster ha un periodo di validità di 10 anni. Le CA successive create tramite il processo di rotazione hanno periodi di validità di 5 anni. Tutte le CA future per il cluster

seguiranno il periodo di validità di 5 anni. Puoi verificare la scadenza della tua CA utilizzando. `describe-certificate-authority`

Posso avviare personalmente una rotazione della CA prima AWS lo fa automaticamente?

Sì. È possibile aggiungere una CA successore in qualsiasi momento utilizzando. `aws eks create-certificate-authority --cluster-name my-cluster` Non è necessario attendere l'inizio della AWS rotazione. Iniziare in anticipo ti dà più tempo per identificare e aggiornare i nodi di lavoro e i clienti esterni secondo i tuoi orari.

Posso eseguire il rollback dopo aver attivato una CA successiva?

Sì, il rollback della CA è disponibile dopo l'attivazione della CA successore purché la finestra di rollback non sia scaduta. CA rollback riattiva la CA precedente come autorità di firma. Non è disponibile dopo l'attivazione automatica finale (45 giorni prima della scadenza della CA). Puoi controllare il `rollbackAvailable` campo sulla tua CA utilizzando`describe-certificate-authority`. Per i dettagli, consulta la sezione CA Rollback.

Come faccio a sapere se è sicuro attivare la CA successiva?

L'attivazione della CA successore è sicura quando tutti i nodi di lavoro gestiti (modalità automatica non EKS, non Fargate) e i client esterni sono stati aggiornati in modo da considerare attendibili la CA successore. È possibile verificarlo confermando che ogni client può comunicare correttamente con il server API utilizzando la configurazione di trust aggiornata. AWS non fornisce un solo indicatore che indichi che tutti i client sono pronti, in quanto non è in grado di visualizzare i sistemi esterni. Iniziate il processo di scoperta in anticipo per avere il tempo di identificare tutti i clienti.

Come posso monitorare l'avanzamento della rotazione della CA su più cluster?

È possibile eseguire questa operazione a livello di programmazione utilizzando la AWS CLI o l'API EKS. Usa per ogni cluster`aws eks list-certificate-authorities`. I seguenti campi forniscono la consapevolezza della situazione per il monitoraggio a livello di flotta:

- `signingStatus`: indica se una CA sta firmando attivamente i certificati (`NOT_USED`,, `ACTIVATING` `IN_USE`
- `distributionStatus`: indica se AWS è stata completata la distribuzione della CA ai componenti gestiti (`IN_PROGRESS`,, `COMPLETE` `FAILED`,`DELETING`)
- `rollbackAvailable`: indica se il rollback della CA è disponibile dopo l'attivazione della CA successore

- `createdBy`/`activatedBy`: distingue tra operazioni avviate dal cliente e operazioni avviate dal cliente (,) AWSCUSTOMER EKS
- `scheduledEvents.firstAutoActivation`/: mostra le prossime date di attivazione automatica `scheduledEvents.finalAutoActivation` AWS

Il seguente script di esempio verifica lo stato di rotazione della CA in un elenco di cluster:

```
#!/bin/bash
CLUSTERS=("cluster-1" "cluster-2" "cluster-3")
REGION="us-west-2"

for CLUSTER in "${CLUSTERS[@]}"; do
  echo "--- $CLUSTER ---"
  aws eks list-certificate-authorities \
    --cluster-name "$CLUSTER" \
    --region "$REGION" \
    --query 'certificateAuthorities[].
{Id:id,Signing:signingStatus,Distribution:distributionStatus,Expiry:validity.notAfter}'
 \
    --output table
done
```

Puoi estenderlo per coprire più regioni e account e filtrare i cluster a cui è stata aggiunta una CA successore, in attesa di intervento o che si stanno avvicinando alle scadenze di attivazione automatica.

Le notifiche vengono inoltre inviate per cluster tramite AWS Health ed e-mail in ogni fase del ciclo di vita della rotazione.

Cosa succede se attivo la CA successiva prima di aggiornare tutti i miei clienti?

Qualsiasi client che non è stato aggiornato per considerare attendibile la CA successore perderà la connettività al cluster EKS. Dopo l'attivazione della CA successore, il server API presenta i certificati firmati dalla CA successore. I client che non si fidano di essa falliranno la verifica TLS e non saranno in grado di connettersi. In tal caso, puoi ripristinare la CA precedente (se la finestra di rollback è ancora disponibile) per ripristinare la connettività mentre correggi i client rimanenti.

Cosa succede se non raggiungo la scadenza della CA?

AWS impedisce che ciò accada. Le protezioni automatiche assicurano che il cluster EKS non raggiunga la scadenza della CA senza una CA valida. Se non l'hai fatto, aggiungeremo una CA

successore e la attiveremo automaticamente circa 6 mesi prima della scadenza. Se effettui il rollback dopo la prima attivazione automatica, AWS esegue un rollforward definitivo 45 giorni prima della scadenza. Il tuo cluster rimarrà disponibile.

Tuttavia, se i nodi di lavoro gestiti (modalità automatica non EKS, non Fargate) e i client esterni non sono stati aggiornati in modo da considerare attendibili la CA successore al momento dell'attivazione automatica, tali componenti perderanno la connettività al server API.

Posso perdere l'accesso al mio cluster durante la rotazione della CA?

Il cluster EKS (piano di controllo) rimane disponibile per l'intero ciclo di vita di rotazione di CA. AWS le misure di protezione assicurano che il cluster stesso non diventi non disponibile.

Tuttavia, i singoli client gestiti possono perdere l'accesso se non vengono aggiornati in modo da considerare attendibili la CA successore prima dell'attivazione della CA successore. Ad esempio, se kubeconfig, CI/CD pipeline o strumento di monitoraggio fanno ancora riferimento solo alla CA in uscita, tali client non saranno in grado di connettersi dopo l'attivazione della CA successore. In tal caso, CA rollback può ripristinare l'accesso mentre si aggiornano i client interessati.

Devo riavviare i miei pod?

L'esecuzione dei workload pod non è direttamente influenzata dalla rotazione delle CA. Il kubelet su ogni nodo gestisce le comunicazioni con il server API, quindi i workload pod continueranno a funzionare senza interruzioni finché i nodi saranno aggiornati. Tuttavia, i controller e gli operatori interni al cluster che utilizzano client-go per comunicare con il server API potrebbero dover essere riavviati dopo l'attivazione della CA successore, poiché client-go non rilegge dinamicamente il CA trust bundle. Per i nodi AWS Fargate, in particolare, gestisce l'aggiornamento automaticamente attraverso il naturale processo di riciclo dei pod. AWS

Il mio pacchetto di fiducia cambierà durante la rotazione?

Sì. Durante la rotazione delle CA, il trust bundle del cluster conterrà contemporaneamente due autorità di certificazione: la CA in uscita e la CA successore. Questo è il comportamento previsto durante il periodo di dual trust ed è il modo in cui il processo di rotazione mantiene la connettività per tutti i componenti.

Le applicazioni e i client devono essere configurati in modo da considerare attendibili un pacchetto CA anziché aggiungere un singolo certificato CA. Il pinning CA (convalida rigorosa rispetto a una singola CA) non è consigliato, poiché causerà errori quando il trust bundle viene aggiornato. Questo vale per qualsiasi configurazione TLS sul lato client che si connette al server API del cluster EKS.

Perché la tempistica delle mie notifiche è diversa da quella descritta in questa documentazione?

Se il cluster è stato creato nel 2018-2019, riceverà notifiche automatiche con una tempistica modificata. La prima notifica includerà le date pertinenti e i passaggi successivi specifici del cluster. I traguardi di notifica standard vengono calcolati in relazione alla data di scadenza della CA del cluster. Per i cluster in questo intervallo, le date calcolate precedono la disponibilità di questa funzionalità, pertanto viene applicata una pianificazione modificata.

## Comprendere ruoli e utenti creati da Amazon EKS tramite RBAC

Quando si crea un cluster Kubernetes, su tale cluster vengono create diverse identità Kubernetes predefinite per il corretto funzionamento di Kubernetes. Amazon EKS crea identità Kubernetes per ciascuno dei suoi componenti predefiniti. Le identità forniscono il controllo degli accessi basato sui ruoli (RBAC) di Kubernetes per i componenti del cluster. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

Quando si installano componenti aggiuntivi opzionali nel cluster, è possibile che vengano aggiunte identità Kubernetes supplementari al cluster. Per ulteriori informazioni sulle identità non trattate in questo argomento, consulta la documentazione del componente aggiuntivo.

Puoi visualizzare l'elenco delle identità Kubernetes create da Amazon EKS sul tuo cluster utilizzando lo strumento Console di gestione AWS o `kubectl` la riga di comando. Tutte le identità degli utenti vengono visualizzate nei registri di `kube` controllo disponibili tramite Amazon. CloudWatch

### Console di gestione AWS

Prerequisito

Il principale IAM da utilizzare deve disporre delle autorizzazioni descritte alla pagina Required permissions.

Per visualizzare le identità create da Amazon EKS utilizzando Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nell'elenco Cluster, seleziona il cluster contenente le identità da visualizzare.
3. Scegli la scheda Risorse.
4. In Resource types (Tipi di risorse), scegli Authorization (Autorizzazione).
5. Scegli ClusterRolesClusterRoleBindings, Ruoli o RoleBindings. Tutte le risorse con il prefisso eks sono create da Amazon EKS. Le risorse di identità supplementari create da Amazon EKS sono:

- La band ClusterRolesi ClusterRoleBindingchiama aws-node. Le risorse aws-node supportano il plugin CNI di Amazon VPC per Kubernetes, che Amazon EKS installa su tutti i cluster.
- Un ClusterRolevpc-resource-controller-role denominato e un vpc-resource-controller-rolebinding denominato. ClusterRoleBinding Queste risorse supportano il Amazon VPC resource controller (Controller risorse Amazon VPC), che Amazon EKS installa su tutti i cluster.

Oltre alle risorse visualizzate nella console, nel cluster esistono le seguenti identità utente speciali, anche se non sono visibili nella configurazione del cluster:

- **`eks:cluster-bootstrap`** : utilizzato per operazioni `kubectl` durante il bootstrap del cluster.
- **`eks:support-engineer`** : utilizzato per le operazioni di gestione del cluster.

6. Scegli una risorsa specifica per visualizzarne i dettagli. Per impostazione predefinita, le informazioni sono mostrate in Vista strutturata. Nell'angolo superiore destro della pagina dei dettagli puoi scegliere la Raw view (Visualizzazione non elaborata) per vedere tutte le informazioni per la risorsa.

## Kubectl

Prerequisito

L'entità che utilizzi (AWS Identity and Access Management (IAM) o OpenID Connect (OIDC)) per elencare le risorse Kubernetes nel cluster deve essere autenticata da IAM o dal tuo provider di identità OIDC. All'entità devono essere concesse autorizzazioni per utilizzare i verbi Kubernetes `get` e `list` per le risorse `Role`, `ClusterRole`, `RoleBinding` e `ClusterRoleBinding` nel cluster che devono essere utilizzate dall'entità. Per ulteriori informazioni sulla concessione alle entità IAM dell'accesso al tuo cluster, consulta the section called "Accesso API Kubernetes". Per ulteriori informazioni sulla concessione dell'accesso al cluster alle entità autenticate dal gestore OIDC, consultare the section called "Collegare il provider OIDC".

**`Per visualizzare le identità create da Amazon EKS utilizzando kubectl`**

Esegui il comando per il tipo di risorsa da visualizzare. Tutte le risorse restituite con il prefisso eks sono create da Amazon EKS. Oltre alle risorse restituite nell'output dai comandi, nel cluster esistono le seguenti identità utente speciali, anche se non sono visibili nella configurazione del cluster:

- **`eks:cluster-bootstrap`** : utilizzato per operazioni `kubectl` durante il bootstrap del cluster.
- **`eks:support-engineer`** : utilizzato per le operazioni di gestione del cluster.

ClusterRoles— `ClusterRoles` sono limitati al tuo cluster, quindi qualsiasi autorizzazione concessa a un ruolo si applica alle risorse in qualsiasi spazio dei nomi Kubernetes sul cluster.

Il comando seguente restituisce tutti i `ClusterRoles` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get clusterroles | grep eks
```

Oltre ai dati `ClusterRoles` restituiti nell'output a cui è preceduto, esistono i seguenti elementi. `eks:ClusterRoles`

- **`aws-node`** : questo `ClusterRole` supporta il plugin CNI di Amazon VPC per Kubernetes, che Amazon EKS installa su tutti i cluster.
- **`vpc-resource-controller-role`** : questo `ClusterRole` supporta il controller di risorse Amazon VPC, che Amazon EKS installa su tutti i cluster.

Per visualizzare le specifiche di a`ClusterRole`, sostituiscilo *`eks:k8s-metrics`* nel comando seguente con un `ClusterRole` restituito nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`ClusterRole`.

```
kubectl describe clusterrole eks:k8s-metrics
```

Di seguito viene riportato un output di esempio.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings— `ClusterRoleBindings` rientrano nell'ambito del cluster.

Il comando seguente restituisce tutti i `ClusterRoleBindings` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get clusterrolebindings | grep eks
```

Oltre ai `ClusterRoleBindings` restituiti nell'output, esistono i seguenti `ClusterRoleBindings`.

- **`aws-node`** : questo `ClusterRoleBinding` supporta il plugin CNI di Amazon VPC per Kubernetes, che Amazon EKS installa su tutti i cluster.
- **`vpc-resource-controller-rolebinding`** : questo `ClusterRoleBinding` supporta il controller di risorse Amazon VPC, che Amazon EKS installa su tutti i cluster.

Per visualizzare la specifica di a`ClusterRoleBinding`, sostituiscila *`eks:k8s-metrics`* nel comando seguente con una `ClusterRoleBinding` restituita nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`ClusterRoleBinding`.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

Di seguito viene riportato un output di esempio.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Roles: i `Roles` sono racchiusi in un namespace Kubernetes. Tutti i `Roles` creati da Amazon EKS rientrano nell'ambito dello spazio nomi `kube-system`.

Il comando seguente restituisce tutti i `Roles` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get roles -n kube-system | grep eks
```

Per visualizzare la specifica di a`Role`, sostituiscila *`eks:k8s-metrics`* nel comando seguente con il nome di a `Role` restituito nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`Role`.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

Di seguito viene riportato un output di esempio.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names              Verbs
  ---------         -----------------  --------------              -----
  daemonsets.apps   []                 [aws-node]                  [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings— `RoleBindings` sono riconducibili a uno spazio dei nomi Kubernetes. Tutti i `RoleBindings` creati da Amazon EKS rientrano nell'ambito dello spazio nomi `kube-system`.

Il comando seguente restituisce tutti i `RoleBindings` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get rolebindings -n kube-system | grep eks
```

Per visualizzare le specifiche di a`RoleBinding`, sostituiscilo *`eks:k8s-metrics`* nel comando seguente con un valore `RoleBinding` restituito nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`RoleBinding`.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

Di seguito viene riportato un output di esempio.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

# Crittografia dei segreti Kubernetes con KMS su cluster esistenti

> **Important**
>
> Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Se si esegue la versione 1.28 o successiva di Kubernetes, i segreti Kubernetes sono protetti con crittografia a busta per impostazione predefinita. Per ulteriori informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

Se abiliti la crittografia dei segreti, i segreti Kubernetes vengono crittografati utilizzando la chiave KMS selezionata. AWS La chiave KMS deve soddisfare le condizioni seguenti:

- Simmetria
- Possibilità di crittografia e decrittografia dei dati
- Creato nella stessa regione del cluster AWS
- Se la chiave KMS è stata creata in un account diverso, il principale IAM dovrà avere accesso alla stessa.

Per ulteriori informazioni, consulta Consentire ai responsabili IAM di altri account di utilizzare una chiave KMS nella Key Management Service Developer Guide.AWS

> **Warning**
>
> Non è possibile disabilitare la crittografia dei segreti dopo averla abilitata. Questa operazione è irreversibile.

eksctl

Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

È possibile abilitare la crittografia in due modi:

- Aggiungere la crittografia al cluster con un singolo comando.

  Per ricrittografare automaticamente i segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption \
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key
```

  Per disattivare la ricrittografia automatica dei segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption \
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- Aggiungi la crittografia al cluster con un file `kms-cluster.yaml`.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws: kms:region-code:account:key/key
```

  Per ricrittografare automaticamente i segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

  Per disattivare la ricrittografia automatica dei segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

  Console di gestione AWS

  a. Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

b. Aprire la Console Amazon EKS.

c. Scegli il cluster a cui aggiungere la crittografia KMS.

d. Scegli la scheda Overview (Panoramica) (selezionata per impostazione predefinita).

e. Scorri verso il basso fino a Secrets encryption (Crittografia dei segreti) e scegli Enable (Abilita).

f. Seleziona una chiave nell'elenco a discesa e fare clic sul pulsante Enable (Abilita). Se non sono elencate chiavi, è necessario crearne una prima. Per ulteriori informazioni, consultare Creazione chiavi

g. Fai clic sul pulsante Confirm (Conferma) per utilizzare la chiave scelta.

AWS CLI

a. Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

b. Associa la configurazione di crittografia segreta al tuo cluster utilizzando il seguente AWS comando CLI. Sostituire i valori di esempio con i propri valori.

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws: kms:region-code:account:key/key"}}]'
```

Di seguito viene riportato un output di esempio.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
```

```
}
```

c. È possibile monitorare lo stato dell'aggiornamento di crittografia con il comando seguente. Usa il `cluster name` specifico e il valore `update ID` restituito nell'output precedente. Quando viene visualizzato lo stato `Successful`, l'aggiornamento è completo.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

Di seguito viene riportato un output di esempio:

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

d. Per verificare che la crittografia sia attivata nel cluster, eseguire il comando `describe-cluster`. La risposta contiene una stringa `EncryptionConfig`.

```
aws eks describe-cluster --region region-code --name my-cluster
```

Dopo aver abilitato la crittografia nel cluster, sarà necessario crittografare tutti i segreti esistenti con la nuova chiave:

> **Note**
>
> È necessario eseguire il comando seguente soltanto se si utilizza `eksctl` e la crittografia automatica dei segreti è disattivata.

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-encryption-timestamp="time value"
```

> **Warning**
>
> Se si abilita la crittografia dei segreti per un cluster esistente e la chiave KMS utilizzata non viene mai eliminata, non sarà possibile ripristinare il cluster. L'eliminazione della chiave KMS metterà definitivamente il cluster in uno stato degradato. Per ulteriori informazioni, consulta Eliminazione delle chiavi AWS KMS.

> **Note**
>
> Per impostazione predefinita, il `create-key` comando crea una chiave KMS di crittografia simmetrica con una politica chiave che fornisce all'amministratore principale dell'account l'accesso alle azioni e alle AWS risorse KMS. Se si desidera ridurre l'ambito delle autorizzazioni, assicurarsi che le azioni `kms:DescribeKey` e `kms:CreateGrant` siano consentite nella policy per il principale che effettua la chiamata all'API `create-cluster`. Per cluster che utilizzano la crittografia a busta KMS, sono necessarie le autorizzazioni `kms:CreateGrant`. La condizione non `kms:GrantIsForAWSResource` è supportata per l' CreateCluster azione e non deve essere utilizzata nelle politiche KMS per controllare `kms:CreateGrant` le autorizzazioni degli utenti che eseguono. CreateCluster

## Utilizzare i segreti di AWS Secrets Manager con i pod di Amazon EKS

Per mostrare i segreti di Secrets Manager e i parametri dall'archivio parametri come file montati in pod di Amazon EKS, è possibile utilizzare AWS Secrets e Configuration Provider (ASCP) per il driver CSI Kubernetes Secrets Store.

Con ASCP, è possibile archiviare e gestire i segreti in Secrets Manager e recuperarli tramite i carichi di lavoro in esecuzione su Amazon EKS. È possibile utilizzare i ruoli e i criteri IAM per limitare l'accesso ai propri segreti a specifici pod Kubernetes in un cluster. ASCP recupera la Pod identity e scambia l'identità per un ruolo IAM. ASCP assume il ruolo IAM del pod e quindi può recuperare i segreti da Secrets Manager autorizzati per tale ruolo.

Se si utilizza la rotazione automatica di Secrets Manager per i propri segreti, è anche possibile utilizzare la funzione di rotation reconciler del Driver CSI di Secrets Store, per assicurarsi di recuperare il segreto più recente da Secrets Manager.

> **Note**
>
> I gruppi di nodi AWS Fargate (Fargate) non sono supportati.

Per ulteriori informazioni, consulta Rotazione dei segreti di Secrets Manager in Amazon EKS nella AWS Guida per l'utente di Secrets Manager.

## Crittografia a busta predefinita per tutti i dati dell'API Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) fornisce la crittografia a busta predefinita di tutti i dati dell'API Kubernetes per i cluster EKS che eseguono Kubernetes versione 1.28 o successiva.

La crittografia a busta protegge i dati archiviati con il server API Kubernetes. Ad esempio, la crittografia a busta si applica alla configurazione del cluster Kubernetes, ad esempio. `ConfigMaps` La crittografia a busta non si applica ai dati sui nodi o sui volumi EBS. EKS in precedenza supportava la crittografia dei segreti Kubernetes, mentre ora la crittografia a busta si estende a tutti i dati dell'API Kubernetes.

Ciò fornisce un'esperienza predefinita gestita che implementa una difesa approfondita per le applicazioni Kubernetes e non richiede alcuna azione lato utente.

Amazon EKS utilizza AWS Key Management Service (KMS) con il provider Kubernetes KMS v2 per questo ulteriore livello di sicurezza con una chiave di proprietà di Amazon Web Services e la possibilità di portare la propria chiave gestita dal cliente (CMK) da KMS. AWS

### Comprensione della crittografia a busta

La crittografia delle buste è il processo di crittografia dei dati in testo semplice con una chiave di crittografia dei dati (DEK) prima che vengano inviati al datastore (etcd) e quindi di crittografia del

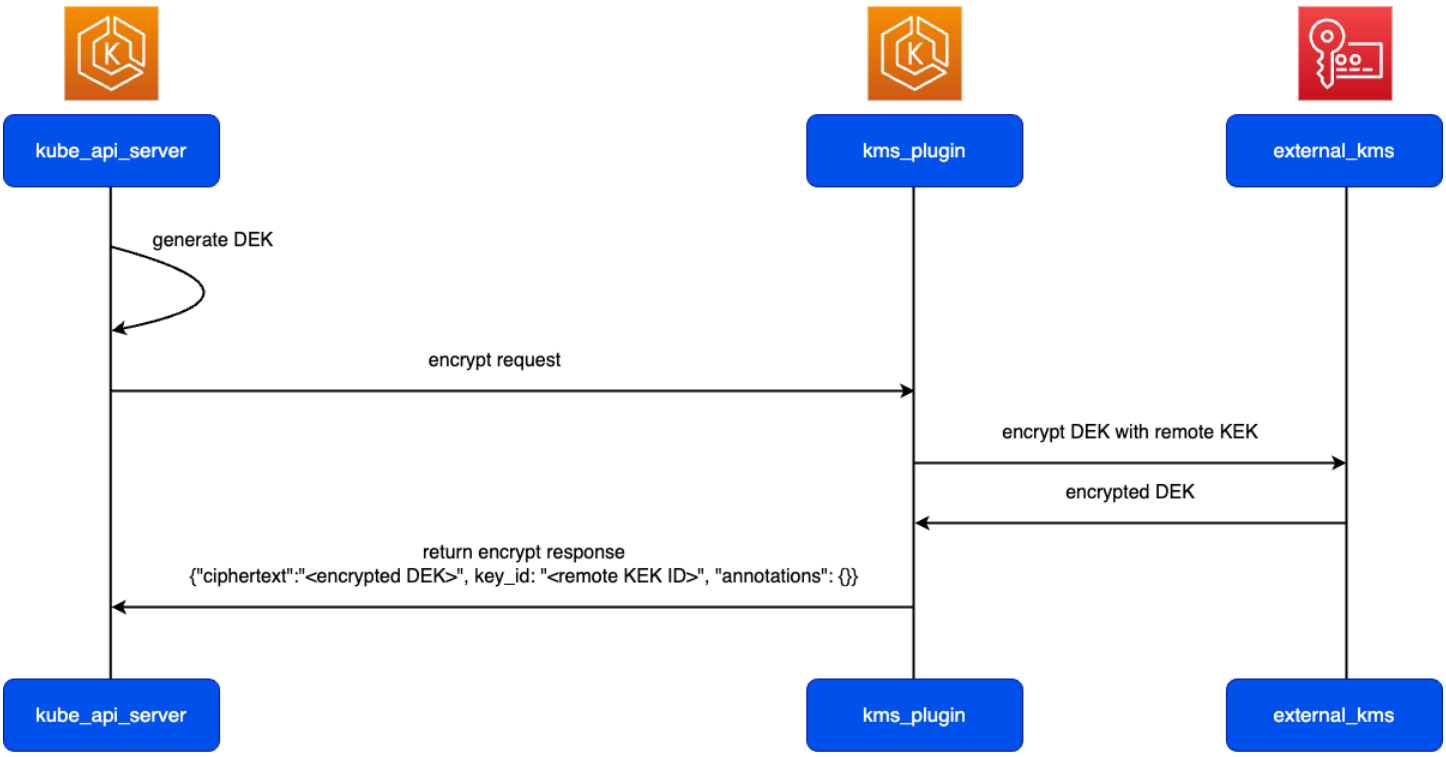
DEK con una chiave KMS principale archiviata in un sistema KMS (AWS KMS) remoto e gestito centralmente. Si tratta di una strategia di difesa approfondita perché protegge i dati con una chiave di crittografia (DEK) e quindi aggiunge un altro livello di sicurezza proteggendo tale DEK con una chiave di crittografia separata e archiviata in modo sicuro, chiamata chiave di crittografia della chiave (KEK).

## In che modo Amazon EKS abilita la crittografia predefinita delle buste con KMS v2 e AWS KMS

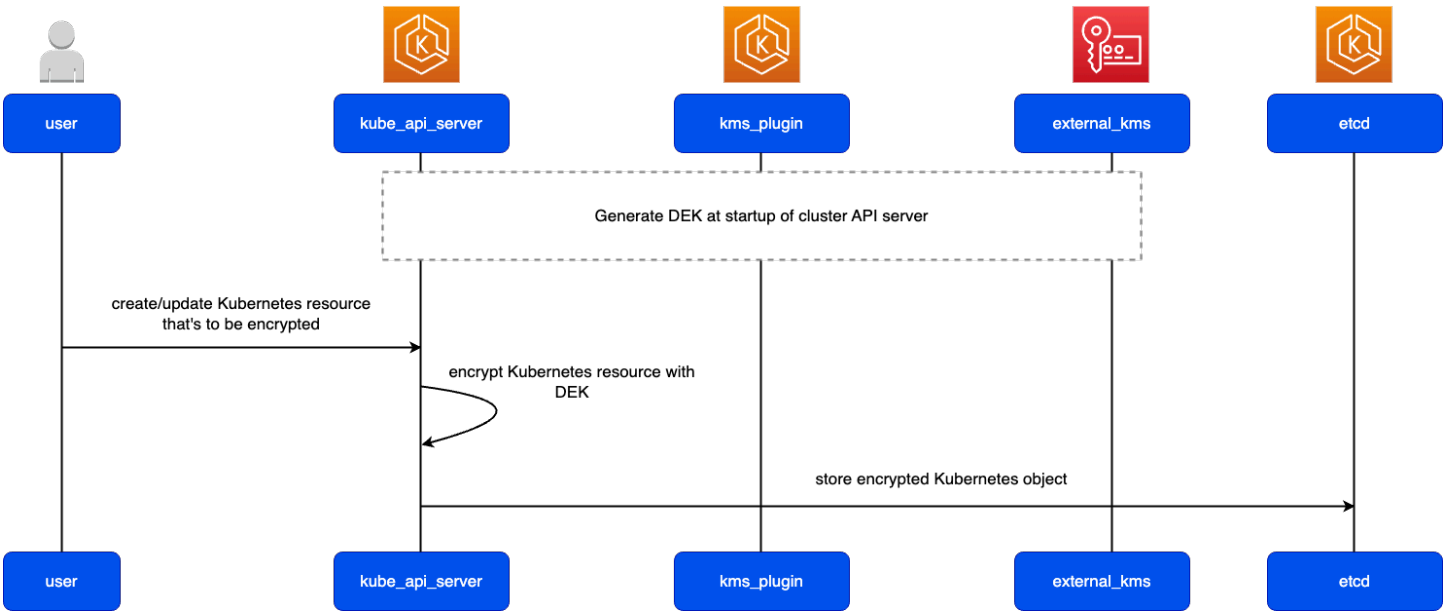
Amazon EKS utilizza KMS v2 per implementare la crittografia a busta predefinita per tutti i dati dell'API nel piano di controllo (control-plane) Kubernetes gestito prima che vengano conservati nel database etcd. All'avvio, il server API del cluster genera una chiave di crittografia dei dati (DEK) da un seed segreto combinato con dati generati casualmente. Inoltre, all'avvio, il server API effettua una chiamata al plug-in KMS per crittografare il seme DEK utilizzando una chiave di crittografia a chiave remota (KEK) di KMS. AWS Si tratta di una chiamata una tantum eseguita all'avvio del server API e durante la rotazione KEK. Il server API memorizza quindi nella cache il seed della DEK crittografato. Successivamente, il server API utilizza il seed della DEK memorizzato nella cache per generare altre DEK monouso basate su una funzione di derivazione della chiave (KDF). Ciascuna di queste DEK generate viene quindi utilizzata solo una volta per crittografare una singola risorsa Kubernetes prima che venga archiviata in etcd. Con l'uso di un seed della DEK crittografato e memorizzato nella cache in KMS v2, il processo di crittografia delle risorse Kubernetes nel server API è sia più performante che conveniente.

Per impostazione predefinita, questa KEK è di proprietà di AWS, ma puoi facoltativamente importarne una da KMS. AWS

Il diagramma riportato di seguito mostra la generazione e la crittografia di una DEK all'avvio del server API.




Il diagramma di alto livello riportato di seguito mostra la crittografia di una risorsa Kubernetes prima che venga archiviata in etcd.



## Domande frequenti

In che modo la crittografia a busta predefinita migliora il livello di sicurezza del mio cluster EKS?

Questa funzionalità riduce la superficie e il periodo di tempo in cui i metadati e i contenuti dei clienti non vengono crittografati. Con la crittografia a busta predefinita, i metadati e i contenuti dei clienti sono solo temporaneamente non crittografati nella memoria di kube-apiserver prima di essere archiviati in etcd. La memoria di kube-apiserver è protetta tramite il sistema Nitro. Amazon EKS utilizza solo istanze Nitro-based EC2 per il piano di controllo Kubernetes gestito. Queste istanze hanno un design di controllo di sicurezza che impediscono a qualsiasi sistema o persona di accedere alla memoria.

Quale versione di Kubernetes devo eseguire per avere questa funzionalità?

Per abilitare la crittografia a busta predefinita, il cluster Amazon EKS deve eseguire Kubernetes versione 1.28 o successiva.

I miei dati sono comunque sicuri se utilizzo una versione del cluster Kubernetes che non supporta questa funzionalità?

Sì. In AWS, la sicurezza è la nostra massima priorità. Basiamo tutta la nostra trasformazione e innovazione digitale sulle pratiche operative di sicurezza più elevate e restiamo impegnati a innalzare tale livello.

Tutti i dati archiviati in etcd sono crittografati a livello di disco per ogni cluster EKS, indipendentemente dalla versione di Kubernetes in esecuzione. EKS utilizza chiavi root che generano chiavi di crittografia del volume gestite dal servizio EKS. Inoltre, ogni cluster Amazon EKS viene eseguito in un VPC isolato utilizzando macchine virtuali specifiche del cluster. Grazie a questa architettura e alle nostre pratiche in materia di sicurezza operativa, Amazon EKS ha ottenuto diversi livelli e standard di conformità, tra cui l'idoneità SOC 1,2,3 PCI-DSS, ISO e HIPAA. Questi livelli e standard di conformità vengono mantenuti per tutti i cluster EKS con o senza crittografia a busta predefinita.

Come funziona la crittografia a busta in Amazon EKS?

All'avvio, il server API del cluster genera una chiave di crittografia dei dati (DEK) da un seed segreto combinato con dati generati casualmente. Inoltre, all'avvio, il server API effettua una chiamata al plug-in KMS per crittografare il DEK utilizzando una chiave di crittografia a chiave remota (KEK) di KMS. AWS Si tratta di una chiamata una tantum eseguita all'avvio del server API e durante la rotazione KEK. Il server API memorizza quindi nella cache il seed della DEK crittografato.

Successivamente, il server API utilizza il seed della DEK memorizzato nella cache per generare altre DEK monouso basate su una funzione di derivazione della chiave (KDF). Ciascuna di queste DEK generate viene quindi utilizzata solo una volta per crittografare una singola risorsa Kubernetes prima che venga archiviata in etcd.

È importante notare che sono state effettuate chiamate aggiuntive dal server API per verificare lo stato e la normale funzionalità dell'integrazione KMS. AWS Questi controlli sanitari aggiuntivi sono visibili nel tuo AWS CloudTrail.

Devo fare qualcosa o modificare le autorizzazioni affinché questa funzionalità sia abilitata nel mio cluster EKS?

Non è necessario eseguire alcuna operazione. La crittografia a busta in Amazon EKS è ora una configurazione predefinita abilitata in tutti i cluster che eseguono Kubernetes versione 1.28 o successiva. L'integrazione AWS KMS è stabilita dal server API Kubernetes gestito da. AWS Ciò significa che non è necessario configurare alcuna autorizzazione per iniziare a utilizzare la crittografia KMS per il cluster.

Come posso sapere se la crittografia a busta predefinita è abilitata sul mio cluster?

Se esegui la migrazione per utilizzare la tua CMK, puoi visualizzare l'ARN della chiave KMS associata al tuo cluster. Inoltre, puoi visualizzare i registri degli AWS CloudTrail eventi associati all'uso della CMK del tuo cluster.

Se il cluster utilizza una chiave AWS di proprietà, questa verrà descritta in dettaglio nella console EKS (escluso l'ARN della chiave).

Può AWS accedere al AWS chiave proprietaria utilizzata per la crittografia delle buste predefinita in Amazon EKS?

No. AWS dispone di rigorosi controlli di sicurezza in Amazon EKS che impediscono a chiunque di accedere a qualsiasi chiave di crittografia in testo semplice utilizzata per proteggere i dati nel database etcd. Queste misure di sicurezza vengono applicate anche alla chiave KMS proprietaria. AWS

La crittografia a busta predefinita è abilitata nel mio cluster EKS esistente?

Se esegui il cluster Amazon EKS con Kubernetes versione 1.28 o successiva, la crittografia a busta predefinita di tutti i dati dell'API Kubernetes è abilitata. Per i cluster esistenti, Amazon EKS utilizza l'`eks:kms-storage-migrator`RBAC ClusterRole per migrare dati che in precedenza non erano crittografati in busta in etcd verso questo nuovo stato di crittografia.

Cosa significa se ho già abilitato la crittografia a busta per i segreti nel mio cluster EKS?

Se disponi di una chiave gestita dal cliente (CMK) in KMS che è stata utilizzata per crittografare a busta i tuoi segreti Kubernetes, quella stessa chiave verrà utilizzata come KEK per la crittografia a busta di tutti i tipi di dati dell'API Kubernetes nel cluster.

La gestione di un cluster EKS con crittografia a busta predefinita comporta costi aggiuntivi?

Non ci sono costi aggiuntivi associati al piano di controllo (control-plane) Kubernetes gestito se utilizzi una chiave di proprietà di Amazon Web Services per la crittografia a busta predefinita. Per impostazione predefinita, ogni cluster EKS che esegue Kubernetes versione 1.28 o successiva utilizza una chiave di proprietà di Amazon Web Services. Tuttavia, se utilizzi la tua chiave AWS KMS, verranno applicati i normali prezzi KMS.

Quanto costa usare la mia AWS Chiave KMS per crittografare i dati dell'API Kubernetes nel mio cluster?

Paghi 1 USD al mese per archiviare qualsiasi chiave personalizzata creata o importata in KMS. Addebiti KMS per le richieste di crittografia e decrittografia. È disponibile un piano gratuito di 20.000 richieste al mese per account e paghi 0,03 USD per 10.000 richieste superiori al piano gratuito al mese. Ciò si applica a tutti gli utilizzi del KMS per un account, quindi il costo dell'utilizzo della tua chiave AWS KMS sul cluster sarà influenzato dall'utilizzo di questa chiave su altri cluster o risorse all'interno del tuo account. AWS

Gli addebiti di KMS saranno più alti ora che la mia chiave gestita dal cliente (CMK) viene utilizzata per crittografare a busta tutti i dati dell'API Kubernetes oltre ai segreti?

No. La nostra implementazione con KMS v2 riduce significativamente il numero di chiamate effettuate a AWS KMS. Ciò a sua volta ridurrà i costi associati alla CMK indipendentemente dal fatto che i dati Kubernetes aggiuntivi vengano crittografati o decrittografati nel cluster EKS.

Come spiegato in precedenza, il seed della DEK generato utilizzato per la crittografia delle risorse Kubernetes viene archiviato localmente nella cache del server API Kubernetes dopo essere stato crittografato con la KEK remota. Se il seed DEK crittografato non si trova nella cache del server API, il server API chiamerà AWS KMS per crittografare il seme DEK. Il server API memorizza quindi nella cache il seed della DEK crittografato per utilizzi futuri nel cluster senza chiamare KMS. Allo stesso modo, per le richieste di decrittografia, il server API chiamerà AWS KMS per la prima richiesta di decrittografia, dopodiché il seme DEK decrittografato verrà memorizzato nella cache e utilizzato per future operazioni di decrittografia.

Per ulteriori informazioni, consulta: KMS v2 Improvements in the Kubernetes Enhancements on. KEP-3299 GitHub

Posso utilizzare la stessa chiave CMK per più cluster Amazon EKS?

Sì. Per utilizzare nuovamente una chiave, puoi collegarla a un cluster nella stessa regione associando l'ARN al cluster durante la creazione. Tuttavia, se utilizzi la stessa CMK per più cluster EKS, devi adottare le misure necessarie per impedire la disabilitazione arbitraria della CMK. In caso contrario, una CMK disabilitata associata a più cluster EKS avrà un ambito di impatto più ampio sui cluster a seconda di tale chiave.

Cosa succede al mio cluster EKS se la mia CMK non è più disponibile dopo aver abilitato la crittografia a busta predefinita?

Se disabiliti una chiave KMS, non può essere utilizzata in nessuna operazione di crittografiariabiliti. Senza l'accesso a una CMK esistente, il server API non sarà in grado di crittografare e rendere persistenti gli oggetti Kubernetes appena creati, nonché di decrittografare qualsiasi oggetto Kubernetes precedentemente crittografato archiviato in etcd. Se la CMK è disabilitata, il cluster verrà immediatamente messo in unhealthy/degraded uno stato, a quel punto non saremo in grado di adempiere al nostro impegno di servizio finché non riabiliterai la CMK associata.

Quando una CMK è disabilitata, riceverai notifiche sullo stato di degrado del tuo cluster EKS e sulla necessità di abilitare nuovamente la CMK entro 30 giorni dalla sua disabilitazione per garantire il corretto ripristino delle risorse del piano di controllo (control-plane) Kubernetes.

Come posso proteggere il mio cluster EKS dall'impatto di una CMK? disabled/deleted

Per proteggere i cluster EKS da tali eventi, gli amministratori della chiave devono gestire l'accesso alle operazioni della chiave KMS utilizzando policy IAM con un principio di privilegio minimo per ridurre il rischio di qualsiasi disabilitazione o eliminazione arbitraria delle chiavi associate ai cluster EKS. Inoltre, puoi impostare un CloudWatch allarme per ricevere una notifica sullo stato della tua CMK.

Il mio cluster EKS verrà ripristinato se abilito nuovamente la CMK?

Per garantire il corretto ripristino del cluster EKS, consigliamo vivamente di abilitare di nuovo la CMK entro i primi 30 giorni dalla sua disabilitazione. Tuttavia, il corretto ripristino del cluster EKS dipenderà anche dal fatto che subisca o meno modifiche sostanziali all'API a causa di un aggiornamento automatico di Kubernetes che può avvenire mentre il cluster è in uno stato. unhealthy/degraded

Perché il mio cluster EKS è in unhealthy/degraded uno stato dopo aver disabilitato la CMK?

Il server API del piano di controllo EKS utilizza una chiave DEK crittografata e memorizzata nella memoria del server API per crittografare tutti gli oggetti durante le create/update operazioni prima che vengano archiviati in etcd. Quando un oggetto esistente viene recuperato da etcd, il server API utilizza la stessa chiave DEK memorizzata nella cache e decrittografa l'oggetto della risorsa Kubernetes. Se disabiliti la CMK, il server API non avrà alcun impatto immediato a causa della chiave DEK archiviata nella cache nella memoria del server API. Tuttavia, quando l'istanza del server API viene riavviata, non avrà un DEK memorizzato nella cache e dovrà chiamare KMS per le operazioni di crittografia e decrittografia. AWS Senza una CMK, questo processo fallirà con un codice di errore KMS_KEY_DISABLED, che impedisce il corretto avvio del server API.

Cosa succede al cluster EKS se elimino la CMK?

L'eliminazione della chiave CMK associata al cluster EKS ne comprometterà lo stato di integrità in modo non ripristinabile. Senza la CMK del cluster, il server API non sarà più in grado di crittografare e rendere persistenti i nuovi oggetti Kubernetes, nonché di decrittografare gli oggetti precedentemente crittografati archiviati nel database etcd. Dovresti procedere con l'eliminazione di una chiave CMK per il cluster EKS solo quando hai la certezza di non dover più utilizzare il cluster EKS.

Tieni presente che se la CMK non viene trovata (KMS_KEY_NOT_FOUND) o le concessioni per la CMK associata al cluster vengono revocate (KMS_GRANT_REVOKED), il cluster non sarà ripristinabile. Per ulteriori informazioni sullo stato del cluster e sui codici di errore, consulta Domande frequenti sullo stato del cluster e codici di errore con percorsi di risoluzione.

Mi verrà comunque addebitato il costo di un cluster degraded/unhealthy EKS perché ho disabilitato o eliminato la mia CMK?

Sì. Sebbene il piano di controllo EKS non sia utilizzabile in caso di CMK disabilitato, AWS continuerà a utilizzare risorse infrastrutturali dedicate allocate al cluster EKS fino a quando non verrà eliminato dal cliente. Inoltre, il nostro impegno di servizio non si applicherà in tali circostanze perché si tratterà di un'operazione volontaria o passiva da parte del cliente che impedisce il normale stato di integrità e il funzionamento del cluster EKS.

Il mio cluster EKS può essere aggiornato automaticamente quando si trova in unhealthy/degraded uno stato a causa di una CMK disabilitata?

Sì. Tuttavia, se il tuo cluster ha una CMK disabilitata, avrai a disposizione un periodo di 30 giorni per abilitarla nuovamente. In questo periodo di 30 giorni, il cluster Kubernetes non verrà aggiornato

automaticamente. Tuttavia, se questo periodo scade e non hai abilitato nuovamente la CMK, il cluster verrà aggiornato automaticamente alla versione successiva (n+1) con supporto standard, seguendo il ciclo di vita della versione Kubernetes in EKS.

Ti consigliamo vivamente di riattivare rapidamente una CMK disabilitata quando vieni a conoscenza di un cluster interessato. Tieni presente che, sebbene EKS aggiorni automaticamente questi cluster interessati, non è garantito che vengano ripristinati correttamente, soprattutto se il cluster viene sottoposto a più aggiornamenti automatici, poiché ciò potrebbe includere modifiche all'API Kubernetes e comportamenti imprevisti nel processo di bootstrap del server API.

Posso utilizzare un alias della chiave KMS?

Sì. Amazon EKS supporta l'utilizzo di alias delle chiavi KMS. Un alias è un nome descrittivo per una chiave Amazon Web Service KMS. Ad esempio, un alias ti consente di fare riferimento a una chiave KMS come my-key invece di **`1234abcd-12ab-34cd-56ef-1234567890ab`**.

Posso comunque eseguire il backup e il ripristino delle risorse del cluster utilizzando la mia soluzione di backup Kubernetes?

Sì. Puoi utilizzare una soluzione di backup Kubernetes (come Velero) per il disaster recovery, la migrazione e la protezione dei dati del cluster Kubernetes. Se esegui una soluzione di backup Kubernetes che accede alle risorse del cluster tramite il server API, tutti i dati recuperati dall'applicazione verranno decrittografati prima di raggiungere il client. Ciò consentirà di ripristinare le risorse del cluster in un altro cluster Kubernetes.

# Harden Kubernetes RBAC in Amazon EKS

Il controllo degli accessi basato sui ruoli (RBAC) di Kubernetes controlla le azioni che le identità possono eseguire all'interno di un cluster. Molti componenti del cluster, inclusi i driver CSI e altri componenti aggiuntivi installati come, richiedono autorizzazioni ampie per funzionare. DaemonSets La revisione e l'ambito di queste autorizzazioni riducono la portata potenziale di qualsiasi accesso non intenzionale.

Questo argomento descrive le considerazioni sulle autorizzazioni per i componenti comuni del cluster e i controlli consigliati.

## DaemonSet autorizzazioni per gli account di servizio

DaemonSet I pod vengono eseguiti su ogni nodo del cluster, quindi i relativi token dell'account di servizio e le autorizzazioni RBAC concesse da tali token sono presenti su ogni nodo.

Un processo non autorizzato su un nodo può essere in grado di accedere ai token dell'account di servizio di altri Pod in esecuzione sullo stesso nodo, inclusi i Pod. DaemonSet Le autorizzazioni RBAC concesse agli account di DaemonSet servizio sono le stesse su ogni nodo del cluster.

I componenti comunemente distribuiti includono: DaemonSets

- driver di nodi CSI (`ebs-csi-node`,,`efs-csi-node`) `mountpoint-s3-csi-node`
- Il plug-in Amazon VPC CNI () `aws-node`
- `kube-proxy`

Se un DaemonSet Pod dispone di credenziali AWS IAM tramite EKS Pod Identity o IAM Roles for Service Accounts (IRSA), anche un processo che ottiene l'accesso all'esterno del contenitore sullo stesso nodo può accedere a tali credenziali. Ciò estende l'ambito di impatto oltre Kubernetes RBAC a qualsiasi AWS autorizzazione API concessa al ruolo IAM di a. DaemonSet

> **Important**
>
> Quando rivedi le autorizzazioni, considera le autorizzazioni RBAC di Kubernetes e le autorizzazioni IAM di ogni account di servizio come accessibili da ogni nodo del cluster. DaemonSet

## Ambito RBAC del driver CSI

I driver CSI generalmente godono di ampie sovvenzioni RBAC perché interagiscono con nodi, volumi persistenti e sistemi di storage. APIs

Autorizzazioni per gli oggetti del nodo

I driver CSI possono richiedere le autorizzazioni RBAC per modificare gli oggetti Node per supportare funzionalità come la rimozione delle contaminazioni o altre attività di gestione dei nodi. A causa delle limitazioni RBAC di Kubernetes, queste autorizzazioni si applicano a tutti gli oggetti Node del cluster, non solo al nodo locale su cui è in esecuzione il driver.

Per il driver CSI EBS, il grafico Helm fornisce un parametro (`node.serviceAccount.disableMutation`) che rimuove l'autorizzazione di modifica del nodo dall'account del servizio. `ebs-csi-node` L'attivazione di questa opzione disabilita la funzione di rimozione delle macchie.

### Esposizione dei token dell'account di servizio

I driver CSI Pods possono utilizzare i token degli account di servizio previsti per l'autenticazione. Su un nodo in cui un processo non autorizzato ha ottenuto l'accesso all'esterno del contenitore, tali token possono essere accessibili tramite il filesystem del contenitore o l'API kubelet. Se l'account di servizio è associato anche a un ruolo IAM tramite EKS Pod Identity o IRSA, è possibile utilizzare un token esposto per ottenere le credenziali IAM. AWS

## Controlli consigliati

### Ambita RBAC al privilegio minimo

- Controlla il collegamento agli ClusterRoles account dei driver e dei servizi CSI. DaemonSet Rimuovi le autorizzazioni non necessarie per i tuoi carichi di lavoro.
- Per il driver EBS CSI, imposta su `node.serviceAccount.disableMutation` `true` se non utilizzi la funzione di rimozione delle macchie.
- Utilizzato per verificare `kubectl auth can-i --list --as=system:serviceaccount:NAMESPACE:SERVICE_ACCOUNT` le autorizzazioni valide.

### Applica gli standard di sicurezza dei Pod

Applica gli standard di sicurezza Kubernetes Pod utilizzando il controller Pod Security Admission integrato o un motore di policy. Come minimo, applica il profilo a livello di cluster e il `baseline` profilo per i namespace dei carichi di lavoro. `restricted` Ciò limita la possibilità di creare contenitori privilegiati al di fuori dei namespace di sistema.

### Usare le policy di rete

Applica politiche di rete per limitare l'uscita dai driver e dai DaemonSet Pod CSI solo agli endpoint di cui hanno bisogno (ad esempio, il server API Kubernetes e gli endpoint di servizio). AWS Ciò riduce la gamma di azioni possibili.

### Monitora l'attività RBAC

Abilita la registrazione degli audit di Kubernetes e monitora eventuali chiamate API impreviste dagli account di servizio. DaemonSet Cerca:

- Modifiche ai nodi dagli account del servizio driver CSI
- Creazione di pod nei namespace di sistema
- Insolito `get` o richiama Secrets `list`

Per ulteriori informazioni, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

# Considerazioni sulla sicurezza per la modalità automatica di Amazon EKS

Questo argomento descrive l'architettura di sicurezza, i controlli e le best practice per la modalità automatica di Amazon EKS. Man mano che le organizzazioni implementano applicazioni containerizzate su scala, mantenere una solida posizione di sicurezza diventa sempre più complesso. EKS Auto Mode implementa controlli di sicurezza automatizzati e si integra con i servizi AWS di sicurezza per aiutarti a proteggere l'infrastruttura, i carichi di lavoro e i dati del cluster. Attraverso funzionalità di sicurezza integrate come la gestione forzata del ciclo di vita dei nodi e l'implementazione automatica delle patch, modalità automatica di EKS ti aiuta a rispettare le best practice di sicurezza riducendo al contempo il sovraccarico operativo.

Prima di procedere con questo argomento, assicurati di conoscere i concetti base della modalità automatica di EKS e di aver esaminato i prerequisiti per abilitare modalità automatica di EKS sui cluster. Per informazioni generali sulla sicurezza di Amazon EKS, consulta Sicurezza.

La modalità automatica di Amazon EKS si fonda sulle basi di sicurezza esistenti di Amazon EKS, introducendo al contempo controlli di sicurezza automatizzati aggiuntivi per le istanze gestite EC2.

## Sicurezza e autenticazione delle API

La modalità automatica di Amazon EKS utilizza meccanismi di sicurezza della AWS piattaforma per proteggere e autenticare le chiamate all'API Amazon EKS.

- L'accesso all'API Kubernetes è protetto tramite voci di accesso EKS, che si integrano con le identità IAM. AWS
  - Per ulteriori informazioni, consulta the section called "Voci di accesso".
- I clienti possono implementare un controllo granulare degli accessi all'endpoint dell'API Kubernetes tramite la configurazione delle voci di accesso EKS.

## Sicurezza di rete

La modalità automatica di Amazon EKS supporta più livelli di sicurezza di rete:

- Integrazione con VPC
  - Funziona all'interno del cloud privato virtuale (VPC) di Amazon
  - Supporta configurazioni VPC personalizzate e layout di sottorete
  - Abilita la rete privata tra i componenti del cluster
  - Per ulteriori informazioni, consulta Managing security responsibilities for Amazon Virtual Private Cloud
- Policy di rete
  - Supporto nativo per le policy di rete di Kubernetes
  - Capacità di definire regole granulari del traffico di rete
  - Per ulteriori informazioni, consulta the section called "Policy di Kubernetes"

## Sicurezza delle istanze gestite di EC2

La modalità automatica di Amazon EKS opera su istanze gestite EC2 con i seguenti controlli di sicurezza:

### Sicurezza EC2

- Le istanze gestite di EC2 mantengono le funzionalità di sicurezza di Amazon EC2.
- Per ulteriori informazioni sulle istanze gestite da EC2, consulta Security in Amazon EC2.

### Gestione del ciclo di vita delle istanze

Le istanze gestite EC2 gestite da EKS Auto Mode hanno una durata massima di 21 giorni. La modalità automatica di Amazon EKS termina automaticamente le istanze che superano questa durata. Questo limite del ciclo di vita aiuta a prevenire variazioni di configurazione e mantiene il livello di sicurezza.

### Protezione dei dati

- Amazon EC2 Instance Storage è crittografata; questa archiviazione è allegata direttamente all'istanza. Per ulteriori informazioni, consulta Data protection in Amazon EC2.
- La modalità automatica di Amazon EKS gestisce i volumi collegati alle istanze EC2 al momento della creazione, compresi i volumi root e di dati. La modalità automatica di EKS non gestisce completamente i volumi EBS creati utilizzando le funzionalità di storage persistente di Kubernetes.

### Gestione delle patch

- La modalità automatica di Amazon EKS applica automaticamente le patch alle istanze gestite.
- Le patch includono:
  - Aggiornamenti del sistema operativo
  - Patch di sicurezza
  - Componenti della modalità automatica di Amazon EKS

> **Note**
>
> I clienti hanno la responsabilità di proteggere e aggiornare i carichi di lavoro in esecuzione su queste istanze.

### Controlli sugli accessi

- L'accesso diretto alle istanze è limitato:
  - L'accesso SSH non è disponibile.
  - AWS L'accesso a Systems Manager Session Manager (SSM) non è disponibile.
- Le operazioni di gestione vengono eseguite tramite l'API Amazon EKS e l'API Kubernetes.

## Gestione automatizzata delle risorse

La modalità automatica di Amazon EKS non gestisce completamente i volumi di Amazon Elastic Block Store (Amazon EBS) creati utilizzando le funzionalità di storage persistente di Kubernetes. Inoltre, modalità automatica di EKS non gestisce il bilanciatore del carico elastico (ELB). La modalità automatica di Amazon EKS automatizza le attività di routine per queste risorse.

### Sicurezza dell'archiviazione

- AWS consiglia di abilitare la crittografia per i volumi EBS forniti dalle funzionalità di storage persistente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crea StorageClass".
- Crittografia inattiva tramite KMS AWS

- Puoi configurare il tuo AWS account per applicare la crittografia dei nuovi volumi EBS e delle copie istantanee che crei. Per ulteriori informazioni, consulta Enable Amazon EBS encryption by default nella Guida per l'utente di Amazon EBS.
- Per ulteriori informazioni, consulta Security in Amazon EBS.

### Sicurezza del bilanciatore del carico

- Configurazione automatizzata dei bilanciatori del carico elastico
- SSL/TLS gestione dei certificati tramite l'integrazione AWS di Certificate Manager
- Automazione dei gruppi di sicurezza per il controllo degli accessi al bilanciatore del carico
- Per ulteriori informazioni, consulta Security in Elastic Load Balancing.

## Best practice di sicurezza

La sezione seguente descrive le best practice di sicurezza per la modalità automatica di Amazon EKS.

- Esamina regolarmente le politiche AWS IAM e le voci di accesso EKS.
- Implementa schemi di accesso con privilegi minimi per i carichi di lavoro.
- Monitora l'attività del cluster tramite AWS CloudTrail Amazon CloudWatch. Per ulteriori informazioni, consultare the section called " AWS CloudTrail" e the section called "Amazon CloudWatch".
- Usa AWS Security Hub per la valutazione del livello di sicurezza.
- Implementa gli standard di sicurezza dei pod appropriati per i carichi di lavoro.

# Considerazioni sulla sicurezza per EKS Capabilities

Questo argomento tratta importanti considerazioni sulla sicurezza per EKS Capabilities, tra cui la configurazione dei ruoli IAM, le autorizzazioni Kubernetes e i modelli architetturali per le implementazioni multi-cluster e la gestione delle risorse tra account. AWS

EKS Capabilities utilizza una combinazione di ruoli IAM, voci di accesso EKS e Kubernetes RBAC per fornire un accesso sicuro ai servizi, alle risorse Kubernetes all'interno del cluster e alle integrazioni con Secrets Manager e altri AWS servizi. AWS CodeConnections AWS AWS

## Capability: ruolo IAM

Quando crei una capacità, fornisci un ruolo di funzionalità IAM che EKS utilizza per eseguire azioni per tuo conto. Questo ruolo deve:

- Avere lo stesso AWS account del cluster e della risorsa di capacità
- Disponi di una politica di fiducia che consenta al responsabile del `capabilities.eks.amazonaws.com` servizio di assumere il ruolo
- Disponi delle autorizzazioni IAM appropriate per il tipo di funzionalità e il caso d'uso, a seconda delle tue esigenze. Per informazioni dettagliate sulle autorizzazioni IAM richieste, consultathe section called " AWS CodeConnections", e the section called " AWS Secrets Manager" the section called "Configurazione delle autorizzazioni "

È consigliabile considerare l'ambito dei privilegi richiesti per il caso d'uso specifico e concedere solo le autorizzazioni necessarie per soddisfare i requisiti. Ad esempio, quando si utilizza EKS Capability for Kube Resource Orchestrator, potrebbero non essere richieste autorizzazioni IAM, mentre quando si utilizza EKS Capability for AWS Controllers for Kubernetes è possibile concedere l'accesso completo a uno o più servizi. AWS

> **Important**
>
> Sebbene alcuni casi d'uso possano giustificare l'uso di ampi privilegi amministrativi, segui il principio del privilegio minimo concedendo solo le autorizzazioni IAM minime richieste per il tuo caso d'uso specifico, limitando l'accesso a risorse specifiche utilizzando ARN e chiavi di condizione anziché utilizzare autorizzazioni wildcard.

Per informazioni dettagliate sulla creazione e la configurazione dei ruoli IAM di funzionalità, consulta. the section called "Capacità, ruolo IAM"

## Voci di accesso EKS

Quando crei una funzionalità con un ruolo IAM, Amazon EKS crea automaticamente una voce di accesso per quel ruolo nel tuo cluster. Questa voce di accesso concede alla funzionalità le autorizzazioni di base di Kubernetes per funzionare.

> **Note**
>
> Le voci di accesso vengono create per il cluster in cui viene creata la funzionalità. Per le distribuzioni di Argo CD su cluster remoti, è necessario creare voci di accesso su tali cluster con le autorizzazioni appropriate affinché la funzionalità Argo CD possa distribuire e gestire le applicazioni.

La voce di accesso include:

- L'IAM ha il ruolo di ARN come principale
- Capability-specific politiche di accesso che concedono le autorizzazioni Kubernetes di base
- Ambito appropriato (a livello di cluster o con ambito di namespace) in base al tipo di funzionalità

> **Note**
>
> Per Argo CD, le autorizzazioni con ambito di namespace vengono concesse allo spazio dei nomi specificato nella configurazione della capacità (il valore predefinito è). `argocd`

Criteri di accesso predefiniti per capacità

Ogni tipo di funzionalità concede al ruolo di funzionalità le autorizzazioni richieste, impostando diverse politiche di accesso predefinite come segue:

kro

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSKROPolicy`(con ambito cluster)

  Concede le autorizzazioni per guardare, gestire ResourceGraphDefinitions e creare istanze di risorse personalizzate definite da RGD.

ACK

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSACKPolicy`(con ambito cluster)

  Concede le autorizzazioni per creare, leggere, aggiornare ed eliminare risorse personalizzate ACK in tutti i namespace.

CD Argo

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy`(con ambito cluster)

  Concede autorizzazioni a livello di cluster ad Argo CD per scoprire risorse e gestire oggetti con ambito di cluster.

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy`(con ambito di namespace)

  Concede autorizzazioni a livello di namespace per Argo CD per distribuire e gestire le applicazioni. L'ambito è basato sullo spazio dei nomi specificato nella configurazione della funzionalità (il valore predefinito è). `argocd`

Per informazioni più dettagliate, vederethe section called "Rivedere le policy di accesso".

## Autorizzazioni Kubernetes aggiuntive

Alcune funzionalità potrebbero richiedere autorizzazioni Kubernetes aggiuntive oltre alle politiche di immissione di accesso predefinite. Puoi concedere queste autorizzazioni utilizzando:

- Politiche di accesso: associa politiche gestite aggiuntive alla voce di accesso
- Kubernetes RBAC: creazione `Role` di `ClusterRole` collegamenti per l'utente Kubernetes della funzionalità

Autorizzazioni dei lettori segreti ACK

Alcuni controller ACK devono leggere i segreti di Kubernetes per recuperare dati sensibili come le password dei database. I seguenti controller ACK richiedono un accesso segreto in lettura:

- `acm`, `acmpca`, `documentdb`, `memorydb`, `mq`, `rds`, `secretsmanager`

Per concedere autorizzazioni di lettura segrete:

1. `arn:aws:eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy`Associa la politica di accesso alla voce di accesso della funzionalità
2. Ambita la policy a namespace specifici in cui le risorse ACK faranno riferimento ai segreti o concederanno l'accesso a livello di cluster

> **Important**
>
> Le autorizzazioni di lettura segrete sono limitate ai namespace specificati quando si associa la politica di immissione degli accessi. Ciò consente di limitare i segreti a cui la funzionalità può accedere.

o autorizzazioni arbitrarie per le risorse

Per impostazione predefinita, kro può controllare e gestire ResourceGraphDefinitions (RGD) e le relative istanze. Per gestire risorse composte come Deployments, Services o ConfigMaps, configura autorizzazioni Kubernetes aggiuntive.

Per concedere a kro le autorizzazioni per creare risorse:

Opzione 1: accedere alle politiche di ingresso

Associa politiche di accesso predefinite come `AmazonEKSAdminPolicy` o `AmazonEKSEditPolicy` alla voce di accesso della funzionalità.

Opzione 2: Kubernetes RBAC

Creane uno `ClusterRoleBinding` che conceda all'utente Kubernetes della funzionalità le autorizzazioni necessarie:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kro-cluster-admin
subjects:
- kind: User
  name: arn:aws:sts::111122223333:assumed-role/my-kro-role/KRO
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

> **Note**
>
> Il nome utente Kubernetes per kro segue lo schema: `arn:aws:sts::ACCOUNT_ID:assumed-role/ROLE_NAME/KRO`

Il nome della sessione `/KRO` (in maiuscolo) viene impostato automaticamente dalla funzionalità EKS kro.

## Autorizzazioni IAM richieste dalla funzionalità

kro (Kube Resource Orchestrator)

Non sono richieste autorizzazioni IAM. È possibile creare un ruolo di capacità senza policy allegate. kro richiede solo le autorizzazioni RBAC di Kubernetes.

ACK AWS (Controller per Kubernetes)

Richiede le autorizzazioni per gestire le AWS risorse che ACK creerà e gestirà. È necessario definire le autorizzazioni per servizi, azioni e risorse specifici in base alle proprie esigenze. Per informazioni dettagliate sulla configurazione delle autorizzazioni ACK, incluse le best practice di produzione con IAM Role Selectors, consulta. the section called "Configurazione delle autorizzazioni "

CD Argo

Per impostazione predefinita, non sono richieste autorizzazioni IAM. Potrebbero essere necessarie autorizzazioni opzionali per:

- AWS Secrets Manager: se si archiviano le credenziali del repository Git in Secrets Manager
- AWS CodeConnections: Se si utilizza CodeConnections per l'autenticazione del repository Git
- Amazon ECR: se si utilizzano grafici Helm archiviati in formato OCI in Amazon ECR

## Best practice di sicurezza

### Privilegio minimo IAM

Concedi alle tue risorse di capacità solo le autorizzazioni necessarie per il tuo caso d'uso. Ciò non significa che non puoi concedere ampie autorizzazioni amministrative alle tue capacità, se necessario. In questi casi, è necessario regolare l'accesso a tali risorse in modo appropriato.

Ruoli di capacità:

- ACK: quando possibile, limita le autorizzazioni IAM a AWS servizi e risorse specifici di cui i tuoi team hanno bisogno, in base al caso d'uso e ai requisiti
- Argo CD: limita l'accesso a specifici repository Git e namespace Kubernetes

- kro: Richiede un ruolo di funzionalità per la policy di fiducia, ma non sono necessarie autorizzazioni IAM (utilizza solo il cluster RBAC)

Esempio: invece di`"Resource": "*"`, specifica i modelli per risorse o gruppi di risorse specifici.

```
"Resource": [
  "arn:aws:s3:::my-app-*",
  "arn:aws:rds:us-west-2:111122223333:db:prod-*"
]
```

Usa le chiavi di condizione IAM per limitare ulteriormente l'accesso:

```
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/Environment": "production"
  }
}
```

Per ulteriori informazioni sulla configurazione IAM, consulta la sezione delle considerazioni per ciascuna funzionalità.

## Isolamento dello spazio dei nomi per i segreti dei CD Argo

La funzionalità Argo CD gestita ha accesso a tutti i segreti di Kubernetes all'interno del namespace configurato (impostazione predefinita:). `argocd` Per mantenere una postura di sicurezza ottimale, segui queste pratiche di isolamento dello spazio dei nomi:

- Mantieni solo i CD-relevant segreti di Argo all'interno dello spazio dei nomi Argo CD
- Evita di archiviare segreti applicativi non correlati nello stesso namespace del CD Argo
- Utilizzate namespace separati per i segreti delle applicazioni che non sono necessari per le operazioni con Argo CD

Questo isolamento garantisce che l'accesso segreto di Argo CD sia limitato alle sole credenziali necessarie per l'autenticazione del repository Git e altre operazioni di Argo. CD-specific

## Kubernetes RBAC

Controlla quali utenti e account di servizio possono creare e gestire risorse di funzionalità. È consigliabile distribuire le risorse di capacità in namespace dedicati con politiche RBAC appropriate.

Esempio: il ruolo RBAC funziona con ACK, consentendo la gestione delle risorse S3 Bucket nel namespace: `app-team`

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ack-s3-manager
  namespace: app-team
rules:
- apiGroups: ["s3.services.k8s.aws"]
  resources: ["buckets"]
  verbs: ["get", "list", "create", "update", "delete"]
```

## Registrazione di controllo

CloudTrail: tutte le operazioni dell'API EKS Capability (creazione, aggiornamento, eliminazione) vengono registrate AWS CloudTrail.

Abilita la CloudTrail registrazione per tracciare:

- Chi ha creato o modificato le funzionalità
- Quando le configurazioni delle funzionalità sono cambiate
- Quali ruoli di funzionalità sono in uso

## Accesso alla rete ed endpoint VPC

Accesso privato all'API Argo CD

È possibile limitare l'accesso al server Argo CD API associando uno o più endpoint VPC all'endpoint Argo CD ospitato. Ciò consente la connettività privata dall'interno del VPC senza attraversare la rete Internet pubblica. L'endpoint VPC fornisce l'accesso sia all'interfaccia utente web di Argo CD che all'API Argo CD (incluso l'accesso alla CLI).

> **Note**
>
> Endpoint VPC connessi agli endpoint API Argo CD ospitati (utilizzando eks-capabilities). *region*.amazonaws.com) non supportano le policy sugli endpoint VPC.

Distribuzione su cluster privati

La funzionalità Argo CD può distribuire applicazioni su cluster EKS completamente privati, fornendo un significativo vantaggio operativo eliminando la necessità di peering VPC o configurazioni di rete complesse. Tuttavia, quando progettate questa architettura, tenete presente che Argo CD estrae la configurazione dai repository Git (che possono essere pubblici) e la applicherà ai cluster privati.

Assicuratevi di:

- Usa repository Git privati per carichi di lavoro sensibili
- Implementa i controlli di accesso e l'autenticazione del repository Git appropriati
- Rivedi e approva le modifiche tramite richieste pull prima di unirle
- Prendi in considerazione l'utilizzo delle finestre di sincronizzazione di Argo CD per controllare quando possono avvenire le distribuzioni
- Monitora i registri di controllo di Argo CD per eventuali modifiche alla configurazione non autorizzate

### Conformità

Le funzionalità EKS sono completamente gestite e dispongono delle certificazioni di conformità di Amazon EKS.

Per informazioni aggiornate sulla conformità, consulta AWS Services in Scope by Compliance Program.

## Fasi successive

- the section called "Configurazione delle autorizzazioni "- Configurare le autorizzazioni IAM per ACK
- the section called "Configurazione delle autorizzazioni "- Configura Kubernetes RBAC per kro
- the section called "Configurazione delle autorizzazioni "- Configura l'integrazione di Identity Center per Argo CD
- the section called "Funzionalità di risoluzione dei problemi"- Risolvi i problemi di sicurezza e autorizzazione

# Identity and Access Management per Amazon EKS

AWS Identity and Access Management (IAM) è AWS un servizio che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse Amazon EKS. IAM è un AWS servizio che puoi utilizzare senza costi aggiuntivi.

## Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Amazon EKS.

Utente del servizio:: se si utilizza il servizio Amazon EKS per svolgere il proprio lavoro, l'amministratore fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di caratteristiche Amazon EKS utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una caratteristica in Amazon EKS, consultare the section called "Risoluzione dei problemi".

Amministratore del servizio: se si è responsabile delle risorse Amazon EKS presso la propria azienda, probabilmente si dispone dell'accesso completo a Amazon EKS. Il compito dell'utente è determinare le caratteristiche e le risorse di Amazon EKS a cui gli utenti del servizio devono accedere. Devi quindi inviare le richieste all'amministratore IAM per modificare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la propria azienda può utilizzare IAM con Amazon EKS, consulta the section called "Amazon EKS e IAM".

Amministratore IAM: se si è amministratore IAM, potrebbe essere interessante ottenere informazioni su come scrivere policy per gestire l'accesso ad Amazon EKS. Per visualizzare policy basate su identità Amazon EKS di esempio che possono essere utilizzate in IAM, consulta the section called "Identity-based politiche".

## Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. È necessario autenticarsi (effettuare l'accesso AWS) come utente root dell' AWS account, come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS Gli utenti di IAM Identity Center (IAM Identity Center), l'autenticazione Single Sign-On

della tua azienda e le tue credenziali Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al Console di gestione AWS o al portale di AWS accesso. Per ulteriori informazioni sull'accesso AWS, vedi Come accedere al tuo AWS account nella Guida per l'utente di AWS accesso.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Signing AWS API request nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) AWS nella Guida per l'utente IAM.

### AWS account (utente root)

Quando si crea un AWS account, si inizia con un'identità di accesso che ha accesso completo a tutti i AWS servizi e le risorse dell'account. Questa identità è denominata utente root dell' AWS account ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

### Utenti e gruppi IAM

Un utente IAM è un'identità all'interno del tuo AWS account che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina

Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni di set di utenti di grandi dimensioni. Ad esempio, potresti avere un gruppo denominato IAMAdminse concedere a quel gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente IAM.

## Ruoli IAM

Un ruolo IAM è un'identità all'interno del tuo AWS account che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in Console di gestione AWS cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato - Per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center.
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento

principale per concedere l'accesso multi-account. Tuttavia, con alcuni AWS servizi, puoi allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

- Accesso tra servizi: alcuni AWS servizi utilizzano le funzionalità di altri AWS servizi. Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, un ruolo di servizio oppure un ruolo collegato al servizio.
  - Sessioni di accesso inoltrato (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama un AWS servizio, in combinazione con il servizio richiedente per effettuare richieste ai AWS servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri AWS servizi o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.
  - Ruolo di servizio - Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio nella Guida per l'utente IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un servizio. AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nell' AWS account e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.
- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un' EC2 istanza e che effettuano richieste AWS CLI AWS o API. Questa soluzione è preferibile alla memorizzazione delle chiavi di accesso all'interno dell'istanza. EC2 Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Usare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella IAM User Guide.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente IAM.

## Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla Console di gestione AWS, dalla AWS CLI o dall' AWS API.

### Policy basate sull'identità

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo AWS account. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente IAM.

## Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o AWS servizi.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

## Elenchi di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportano. ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

## Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai più tipi di policy comuni.

- Limiti delle autorizzazioni - Un limite delle autorizzazioni è una funzionalità avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate sull'identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per maggiori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo del servizio (SCPs): SCPs sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in Organizations AWS . AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più AWS account

di proprietà della tua azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità negli account dei membri, incluso ogni AWS utente root dell'account. Per ulteriori informazioni su Organizations and SCPs, consulta le politiche di controllo dei servizi nella AWS Organizations User Guide.

- Policy di sessione - Le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate sull'identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente IAM.

### Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

## Funzionamento di Amazon EKS con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon EKS, è necessario comprendere quali funzioni IAM sono disponibili per l'uso con Amazon EKS. Per avere una visione di alto livello di come Amazon EKS e altri AWS servizi funzionano con IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Argomenti

- Policy basate su identità Amazon EKS
- Policy basate sulle risorse Amazon EKS
- Autorizzazione basata su tag Amazon EKS
- Ruoli IAM di Amazon EKS

### Policy basate su identità Amazon EKS

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte.

Amazon EKS supporta specifiche operazioni, risorse e chiavi di condizione. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consultare Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Azioni

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Le azioni politiche in genere hanno lo stesso nome dell'operazione AWS API associata. Ci sono alcune eccezioni, ad esempio le operazioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includere le operazioni in una policy per concedere le autorizzazioni per eseguire l'operazione associata.

Le operazioni delle policy in Amazon EKS utilizzano il seguente prefisso prima dell'operazione: `eks:`. Ad esempio, per concedere a qualcuno l'autorizzazione per ottenere le informazioni descrittive su un cluster Amazon EKS, includere l'operazione `DescribeCluster` nella policy. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`.

Per specificare più operazioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

```
"Action": ["eks:action1", "eks:action2"]
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "eks:Describe*"
```

Per visualizzare un elenco di operazioni di Amazon EKS, consulta Operazioni definite da Amazon Elastic Kubernetes Service in Service Authorization Reference.

Resources

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). È possibile eseguire questa operazione per operazioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

La risorsa del cluster Amazon EKS dispone del seguente ARN.

```
 arn:aws: eks:region-code:account-id:cluster/cluster-name
```

Per ulteriori informazioni sul formato di ARNs, consulta Amazon resource names (ARNs) e AWS service namespaces.

Ad esempio, per specificare il cluster con il nome *my-cluster* nell'istruzione, utilizzare il seguente ARN:

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/my-cluster"
```

Per specificare tutti i cluster che appartengono a un account e a una AWS regione specifici, usa il carattere jolly (*):

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/*"
```

Alcune operazioni Amazon EKS, ad esempio quelle utili per la creazione di risorse, non possono essere eseguite su una risorsa specifica. In questi casi, è necessario utilizzare il carattere jolly (*).

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse Amazon EKS e relativi ARNs, consulta Resources defined by Amazon Elastic Kubernetes Service nel Service Authorization Reference. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

Chiavi di condizione

Amazon EKS definisce il proprio set di chiavi di condizione e supporta anche l'uso di alcune chiavi di condizione globali. Per visualizzare tutte le chiavi di condizione AWS globali, consulta AWS Global Condition Context Keys nella IAM User Guide.

È possibile impostare le chiavi di condizione quando si associa un provider OpenID Connect al cluster. Per ulteriori informazioni, consulta the section called "Policy IAM di esempio".

Tutte le operazioni Amazon EC2 supportano le chiavi di condizione `aws:RequestedRegion` e `ec2:Region`. Per ulteriori informazioni, consulta Esempio: limitazione dell'accesso a una AWS regione specifica.

Per un elenco di chiavi di condizione di Amazon EKS, consultare Condizioni per Amazon Elastic Kubernetes Service in Service Authorization Reference. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

Esempi

Per visualizzare esempi di policy basate su identità Amazon EKS, consultare the section called "Identity-based politiche".

Quando si crea un cluster Amazon EKS, il principale IAM che crea il cluster riceve automaticamente le autorizzazioni `system:masters` nella configurazione del controllo degli accessi basato sul ruolo (RBAC) nel piano di controllo di Amazon EKS. Questo principale non è visualizzato in una configurazione visibile qualsiasi, quindi assicurati di tenere traccia di quale principale ha originariamente creato il cluster. Per concedere a ulteriori principali IAM la capacità di interagire con il cluster, modifica `aws-auth ConfigMap` all'interno di Kubernetes e crea `rolebinding` o `clusterrolebinding` Kubernetes con il nome di `group` specificato in `aws-auth ConfigMap`.

Per ulteriori informazioni sull'utilizzo di ConfigMap, vederethe section called "Accesso API Kubernetes".

## Policy basate sulle risorse Amazon EKS

Amazon EKS non supporta policy basate su risorse.

## Autorizzazione basata su tag Amazon EKS

È possibile allegare i tag alle risorse Amazon EKS o inoltrarli in una richiesta ad Amazon EKS. Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy

utilizzando le chiavi di condizione `aws:ResourceTag/key-name` , `aws:RequestTag/key-name` o `aws:TagKeys`. Per ulteriori informazioni sull'assegnazione di tag delle risorse di Amazon EKS, consultare the section called "Tagging delle risorse ". Per ulteriori informazioni sulle operazioni in cui è possibile utilizzare i tag nelle chiavi di condizione, consulta Operazioni definite da Amazon EKSnel manuale Service Authorization Reference.

## Ruoli IAM di Amazon EKS

Un ruolo IAM è un'entità all'interno del tuo AWS account che dispone di autorizzazioni specifiche.

### Utilizzo di credenziali temporanee con Amazon EKS

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. È possibile ottenere credenziali di sicurezza temporanee chiamando operazioni API AWS STS come AssumeRoleo. GetFederationToken

Amazon EKS supporta l'uso di credenziali temporanee.

### Ruoli collegati ai servizi

I ruoli collegati ai AWS servizi consentono ai servizi di accedere alle risorse di altri servizi per completare un'azione per conto dell'utente. I ruoli collegati ai servizi sono visualizzati nell'account IAM e sono di proprietà del servizio. Un amministratore può visualizzare, ma non può modificare le autorizzazioni dei ruoli collegati ai servizi.

Amazon EKS supporta i ruoli collegati ai servizi. Per maggiori dettagli su come creare e gestire i ruoli collegati ai servizi Amazon EKS, consultare the section called "Ruoli collegati ai servizi".

### Ruoli dei servizi

Questa funzionalità consente a un servizio di assumere un ruolo di servizio per conto dell'utente. Questo ruolo consente al servizio di accedere alle risorse in altri servizi per completare un'azione per conto dell'utente. I ruoli dei servizi sono visualizzati nell'account IAM e sono di proprietà dell'account. Ciò significa che un amministratore IAM può modificare le autorizzazioni per questo ruolo. Tuttavia, questo potrebbe pregiudicare la funzionalità del servizio.

Amazon EKS supporta i ruoli del servizio. Per ulteriori informazioni, consultare the section called "Ruolo IAM del cluster" e the section called "Ruolo IAM del nodo".

Scelta di un ruolo IAM in Amazon EKS

Quando crei una risorsa cluster in Amazon EKS, devi scegliere un ruolo per consentire ad Amazon EKS di accedere a diverse altre AWS risorse per tuo conto. Se hai già creato un ruolo di servizio in precedenza, Amazon EKS ti fornisce un elenco di ruoli da scegliere. È importante scegliere un ruolo con policy gestite da Amazon EKS allegate ad esso. Per ulteriori informazioni, consultare the section called "Verifica della presenza di un ruolo del cluster esistente" e the section called "Verifica della presenza di un ruolo di nodo esistente".

## Esempi di policy basate su identità Amazon EKS

Per impostazione predefinita, gli utenti e i ruoli IAM non dispongono dell'autorizzazione per creare o modificare risorse Amazon EKS. Inoltre, non possono eseguire attività utilizzando Console di gestione AWS la AWS CLI o l' AWS API. Un amministratore IAM deve creare policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore deve quindi allegare queste policy a utenti o IAM che richiedono tali autorizzazioni.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consultare Creazione di policy nella scheda JSON nella Guida per l'utente di IAM.

Quando si crea un cluster Amazon EKS, il principale IAM che crea il cluster riceve automaticamente le autorizzazioni `system:masters` nella configurazione del controllo degli accessi basato sul ruolo (RBAC) nel piano di controllo di Amazon EKS. Questo principale non è visualizzato in una configurazione visibile qualsiasi, quindi assicurati di tenere traccia di quale principale ha originariamente creato il cluster. Per concedere a ulteriori principali IAM la capacità di interagire con il cluster, modifica `aws-auth ConfigMap` all'interno di Kubernetes e crea `rolebinding` o `clusterrolebinding` Kubernetes con il nome di `group` specificato in `aws-auth ConfigMap`.

Per ulteriori informazioni sull'utilizzo di ConfigMap, vederethe section called "Accesso API Kubernetes".

Argomenti

- Best practice per le policy
- Utilizzo della console Amazon EKS
- Consentire agli utenti IAM di visualizzare le loro autorizzazioni
- Crea un cluster Kubernetes su AWS Cloud

- Creazione di un cluster Kubernetes locale su un Outpost
- Aggiornare un cluster Kubernetes
- Elencare o descrivere tutti i cluster
- Annullare un aggiornamento del cluster (rollback)

## Best practice per le policy

Identity-based le policy stabiliscono se qualcuno può creare, accedere o eliminare risorse Amazon EKS nel tuo account. Queste azioni possono comportare costi per il tuo AWS account. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere le autorizzazioni agli utenti e ai carichi di lavoro, utilizza le policy AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo account. AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dal AWS cliente specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite un AWS servizio specifico, ad esempio AWS CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente IAM.

- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel tuo AWS account, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per ulteriori informazioni, consulta Configurazione dell'accesso alle MFA-protected API nella IAM User Guide.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

## Utilizzo della console Amazon EKS

Per accedere alla console Amazon EKS, un principale IAM deve disporre di un set di autorizzazioni minimo. Queste autorizzazioni consentono al committente di elencare e visualizzare i dettagli sulle risorse Amazon EKS presenti nel tuo AWS account. Se crei una policy di basata su identità più restrittiva delle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per i principali associati a tale policy.

Per garantire che i principali IAM possano comunque utilizzare la console Amazon EKS, crea una policy con un nome univoco, ad esempio `AmazonEKSAdminPolicy`. Allega la politica ai principali. Per ulteriori informazioni, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

> **Important**
>
> La policy di esempio riportata di seguito consente a un principale di visualizzare le informazioni sulla scheda Configurazione della console. Per visualizzare le informazioni nelle schede Panoramica e Risorse del Console di gestione AWS, il principale necessita anche delle autorizzazioni Kubernetes. Per ulteriori informazioni, consulta the section called "Autorizzazioni richieste".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

Non è necessario consentire le autorizzazioni minime della console per i principali che effettuano chiamate solo alla CLI o all'API. AWS AWS Al contrario, è possibile accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

## Consentire agli utenti IAM di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o a livello di codice utilizzando la CLI o l'API. AWS AWS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

## Crea un cluster Kubernetes su AWS Cloud

Questa policy di esempio include le autorizzazioni minime richieste per creare un cluster Amazon EKS denominato *`my-cluster`* nella regione. *`us-west-2`* AWS Puoi sostituire la AWS regione con la AWS regione in cui desideri creare un cluster. Se viene visualizzato un avviso che indica che le azioni contenute nella policy non supportano le autorizzazioni a livello di risorsa e richiedono l'inserimento di tali autorizzazioni Console di gestione AWS, può essere **`All resources`** tranquillamente ignorato. Se il tuo account dispone già del ruolo *`AWSServiceRoleForAmazonEKS`*, puoi rimuovere l'azione `iam:CreateServiceLinkedRole` dalla policy. Se hai mai creato un cluster Amazon EKS nel tuo account, questo ruolo è già esistente, a meno che sia stato eliminato.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
            "Condition": {
                "ForAnyValue:StringEquals": {
```

```
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        }
    ]
}
```

## Creazione di un cluster Kubernetes locale su un Outpost

Questa policy di esempio include le autorizzazioni minime richieste per creare un cluster locale Amazon EKS denominato *`my-cluster`* in un Outpost nella regione. *`us-west-2`* AWS Puoi sostituire la AWS regione con la AWS regione in cui desideri creare un cluster. Se viene visualizzato un avviso che indica che le azioni contenute nella policy non supportano le autorizzazioni a livello di risorsa e richiedono l'inserimento di tali autorizzazioni Console di gestione AWS, può essere **`All resources`** tranquillamente ignorato. Se il tuo account dispone già del ruolo `AWSServiceRoleForAmazonEKSLocalOutpost`, puoi rimuovere l'azione `iam:CreateServiceLinkedRole` dalla policy. Se hai già creato un cluster locale Amazon EKS su Outpost nel tuo account, questo ruolo esiste già a meno che tu non l'abbia eliminato.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        }
    ]
}
```

## Aggiornare un cluster Kubernetes

Questa politica di esempio include l'autorizzazione minima richiesta per aggiornare un cluster denominato nella regione us-west-2. *my-cluster* AWS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

## Elencare o descrivere tutti i cluster

Questa policy di esempio include le autorizzazioni minime richieste per elencare e descrivere tutti i cluster dell'account. Un responsabile IAM deve essere in grado di elencare e descrivere i cluster per utilizzare il `update-kubeconfig` AWS comando CLI.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

## Annullare un aggiornamento del cluster (rollback)

Per consentire agli utenti di annullare le operazioni di rollback in corso, concedi l'azione. `eks:CancelUpdate` Puoi definire l'ambito utilizzando l'ARN della risorsa o le condizioni basate sui tag.

Risorsa: ARN-based

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "eks:CancelUpdate",
    "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
  }]
}
```

Tag-based:

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [{
    "Effect": "Allow",
    "Action": "eks:CancelUpdate",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "development"
      }
    }
  }]
}
```

Per ulteriori informazioni sul processo completo di rollback del cluster, consultathe section called "Ripristina la versione di Kubernetes".

## Utilizzo di ruoli collegati ai servizi per Amazon EKS

Amazon Elastic Kubernetes Service utilizza service-linked roles di AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono definiti automaticamente da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per conto dell'utente.

Argomenti

- Utilizzo dei ruoli per i cluster Amazon EKS
- Utilizzo dei ruoli per i gruppi di nodi Amazon EKS
- Utilizzo dei ruoli per i profili Fargate in Amazon EKS
- Utilizzo di ruoli per connettere un cluster Kubernetes ad Amazon EKS
- Utilizzo dei ruoli per i cluster locali Amazon EKS su Outpost
- Utilizzo dei ruoli per la dashboard Amazon EKS

### Utilizzo dei ruoli per i cluster Amazon EKS

Amazon Elastic Kubernetes Service AWS utilizza ruoli collegati al servizio Identity and Access Management (IAM) and Access Management. Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon EKS. Service-linked i ruoli sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS i servizi che funzionano con IAM e cerca i servizi con Sì nella colonna dei ruoli. Service-linked  Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Service-linked autorizzazioni di ruolo per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKS`. Il ruolo consente ad Amazon EKS di gestire i cluster nell'account. Le policy allegate consentono al ruolo di gestire le seguenti risorse: interfacce di rete, gruppi di sicurezza, registri e VPC.

> **Note**
>
> Il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKS` è distinto dal ruolo richiesto per la creazione del cluster. Per ulteriori informazioni, consulta the section called "Ruolo IAM del cluster".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKS` considera attendibili i seguenti servizi:

- `eks.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta le autorizzazioni dei Service-linked ruoli nella Guida per l'utente IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un cluster nella AWS CLI o nell' AWS API, Amazon EKS crea il ruolo collegato al servizio per te. Console di gestione AWS

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando viene creato un cluster, Amazon EKS crea di nuovo il ruolo collegato ai servizi automaticamente.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKS`. Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.

2. Nel pannello di navigazione a sinistra selezionare Clusters (Cluster).
3. Se il cluster dispone di gruppi di nodi o profili Fargate, è necessario eliminarli prima di eliminare il cluster. Per ulteriori informazioni, consultare the section called "Elimina" e the section called "Eliminare profili".
4. Sulla pagina Cluster, scegliere il cluster da eliminare e scegliere Elimina.
5. Digitare il nome del cluster nella finestra di conferma eliminazione, quindi scegliere Elimina.
6. Ripetere questa procedura per tutti gli altri cluster nell'account. Attendere che tutte le operazioni di eliminazione siano completate.

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la AWS CLI o l' AWS API per eliminare il ruolo collegato al `AWSServiceRoleForAmazonEKS` servizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

## Utilizzo dei ruoli per i gruppi di nodi Amazon EKS

Amazon EKS utilizza ruoli collegati ai servizi di AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon EKS. Service-linked i ruoli sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS i servizi che funzionano con IAM e cerca i servizi che hanno Sì nella colonna del ruolo. Service-linked Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Service-linked autorizzazioni di ruolo per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSNodegroup`. Il ruolo consente ad Amazon EKS di gestire i gruppi di nodi nell'account. La policy `AWSServiceRoleForAmazonEKSNodegroup` allegata consente al ruolo di gestire le seguenti risorse: gruppi Auto Scaling, gruppi di sicurezza, modelli di avvio e profili dell'istanza IAM. Per ulteriori informazioni, consulta the section called "AWS policy gestita: AWSServiceRoleForAmazonEKSNodegroup".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSNodegroup` considera attendibili i seguenti servizi:

- `eks-nodegroup.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AWSServiceRoleForAmazonEKSNodegroup

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta le autorizzazioni dei Service-linked ruoli nella IAM User Guide.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un gruppo di nodi gestiti nel Console di gestione AWS, nella AWS CLI o nell' AWS API, Amazon EKS crea il ruolo collegato al servizio per te.

> **Important**
>
> Questo ruolo collegato al servizio può apparire nell'account, se è stata completata un'operazione in un altro servizio che utilizza le caratteristiche supportate da questo ruolo. Se stavi utilizzando il servizio Amazon EKS prima del 1° gennaio 2017,

quando ha iniziato a supportare i ruoli collegati ai servizi, Amazon EKS ha creato il AWSServiceRoleForAmazonEKSNodegroup ruolo nel tuo account. Per ulteriori informazioni, consultare Un nuovo ruolo è apparso nel mio account IAM.

Creazione di un ruolo collegato ai servizi in Amazon EKS (AWS API)

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un gruppo di nodi gestiti nel Console di gestione AWS, nella AWS CLI o nell' AWS API, Amazon EKS crea il ruolo collegato al servizio per te.

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando si crea un altro gruppo di nodi gestiti, Amazon EKS crea nuovamente il ruolo collegato al servizio.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSNodegroup`. Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Seleziona la scheda Compute (Calcolo).
4. Nella sezione Gruppi di nodi, scegliere il gruppo di nodi da eliminare.
5. Digitare il nome del gruppo di nodi nella finestra di conferma eliminazione e quindi scegliere Elimina.
6. Ripetere questa procedura per tutti gli altri gruppi di nodi nel cluster. Attendere che tutte le operazioni di eliminazione siano completate.

Eliminazione manuale del ruolo collegato ai servizi

Usa la console IAM, la AWS CLI o l' AWS API per eliminare il ruolo collegato al servizio. `AWSServiceRoleForAmazonEKSNodegroup` Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

## Utilizzo dei ruoli per i profili Fargate in Amazon EKS

Amazon EKS utilizza ruoli collegati ai servizi di AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon EKS. Service-linked i ruoli sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS i servizi che funzionano con IAM e cerca i servizi che hanno Sì nella colonna del ruolo. Service-linked Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Service-linked autorizzazioni di ruolo per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSForFargate`. Il ruolo consente ad Amazon EKS Fargate di configurare la rete VPC necessaria per i pod Fargate. Le policy allegate consentono al ruolo di creare ed eliminare interfacce di rete elastiche e descrivere interfacce e risorse di rete elastiche.

Ai fini dell'assunzione del ruolo, il ruolo collegato al servizio `AWSServiceRoleForAmazonEKSForFargate` considera attendibili i seguenti servizi:

- `eks-fargate.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSForFargateServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta le autorizzazioni dei Service-linked ruoli nella IAM User Guide.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un profilo Fargate nella Console di gestione AWS, nella AWS CLI o nell' AWS API, Amazon EKS crea il ruolo collegato al servizio per te.

> **Important**
>
> Questo ruolo collegato al servizio può apparire nell'account, se è stata completata un'operazione in un altro servizio che utilizza le caratteristiche supportate da questo ruolo. Se stavi utilizzando il servizio Amazon EKS prima del 13 dicembre 2019, quando ha iniziato a supportare i ruoli collegati ai servizi, Amazon EKS ha creato il ruolo nel tuo account.

AWSServiceRoleForAmazonEKSForFargate Per ulteriori informazioni, consultare Un nuovo ruolo è apparso nel mio account IAM.

Creazione di un ruolo collegato ai servizi in Amazon EKS (AWS API)

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un profilo Fargate nel Console di gestione AWS, nella AWS CLI o nell' AWS API, Amazon EKS crea il ruolo collegato al servizio per te.

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando crei un altro profilo Fargate, Amazon EKS crea nuovamente il ruolo collegato al servizio per te.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSForFargate`. Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.

2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Nella pagina Cluster selezionare il cluster.
4. Seleziona la scheda Compute (Calcolo).
5. Se sono presenti profili Fargate nella sezione Fargate profiles (Profili Fargate), selezionare ciascun profilo singolarmente, quindi scegliere Delete (Elimina).
6. Digitare il nome del cluster nella finestra di conferma eliminazione e quindi scegliere Elimina.
7. Ripetere questa procedura per tutti gli altri profili Fargate nel cluster e per ciascun cluster nell'account.

Eliminazione manuale del ruolo collegato ai servizi

Usa la console IAM, la AWS CLI o l' AWS API per eliminare il ruolo collegato al servizio. AWSServiceRoleForAmazonEKSForFargate Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

## Utilizzo di ruoli per connettere un cluster Kubernetes ad Amazon EKS

Amazon EKS utilizza ruoli AWS collegati al servizio Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon EKS. Service-linked i ruoli sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS i servizi che funzionano con IAM e cerca i servizi con Sì nella colonna dei ruoli. Service-linked  Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Service-linked autorizzazioni di ruolo per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSConnector`. Il ruolo consente ad Amazon EKS di collegarsi ai cluster Kubernetes. Le policy allegate consentono al ruolo di gestire le risorse necessarie per connettersi al cluster Kubernetes registrato.

Ai fini dell'assunzione del ruolo, il ruolo collegato al servizio `AWSServiceRoleForAmazonEKSConnector` considera attendibili i seguenti servizi:

- `eks-connector.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSConnectorServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta le autorizzazioni dei Service-linked ruoli nella Guida per l'utente IAM.

Questo ruolo utilizza le autorizzazioni SSM (Systems Manager) per stabilire connessioni sicure e gestire i cluster Kubernetes connessi.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non devi creare manualmente un ruolo collegato ai servizi per collegare un cluster. Quando connetti un cluster nella AWS CLI o nell' AWS API`eksctl`, Amazon EKS crea il ruolo collegato al servizio per te. Console di gestione AWS

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando viene collegato un cluster, Amazon EKS crea di nuovo il ruolo collegato ai servizi automaticamente.

### Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSConnector`. Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

### Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

### Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Nella pagina Cluster selezionare il cluster.
4. Scegli Annulla registrazione, quindi scegli Ok.

### Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la AWS CLI o l' AWS API per eliminare il ruolo collegato al AWSServiceRoleForAmazonEKSConnector servizio. Per ulteriori informazioni, consultare Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

## Utilizzo dei ruoli per i cluster locali Amazon EKS su Outpost

Amazon Elastic Kubernetes Service AWS utilizza ruoli collegati al servizio Identity and Access Management (IAM) and Access Management. Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon EKS. Service-linked i ruoli sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS i servizi che funzionano con IAM e cerca i servizi con Sì nella colonna dei ruoli. Service-linked  Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Service-linked autorizzazioni di ruolo per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSLocalOutpost`. Il ruolo consente ad Amazon EKS di gestire i cluster locali nell'account. Le policy collegate consentono al ruolo di gestire le seguenti risorse: interfacce di rete, gruppi di sicurezza, registri e istanze Amazon EC2.

> **Note**
>
> Il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSLocalOutpost` è distinto dal ruolo richiesto per la creazione del cluster. Per ulteriori informazioni, consulta the section called "Ruolo IAM del cluster".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSLocalOutpost` considera attendibili i seguenti servizi:

- `outposts.eks-local.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSLocalOutpostServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta le autorizzazioni dei Service-linked ruoli nella Guida per l'utente IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un cluster nella AWS CLI o nell' AWS API, Amazon EKS crea il ruolo collegato al servizio per te. Console di gestione AWS

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando viene creato un cluster, Amazon EKS crea di nuovo il ruolo collegato ai servizi automaticamente.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSLocalOutpost`. Dopo aver creato un ruolo collegato al servizio, non è possibile modificarne il nome, perché potrebbero farvi riferimento diverse entità. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, scegli Clusters (Cluster) Amazon EKS.
3. Se il cluster dispone di gruppi di nodi o profili Fargate, è necessario eliminarli prima di eliminare il cluster. Per ulteriori informazioni, consultare the section called "Elimina" e the section called "Eliminare profili".
4. Sulla pagina Cluster, scegliere il cluster da eliminare e scegliere Elimina.
5. Digitare il nome del cluster nella finestra di conferma eliminazione, quindi scegliere Elimina.
6. Ripetere questa procedura per tutti gli altri cluster nell'account. Attendere che tutte le operazioni di eliminazione siano completate.

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la AWS CLI o l' AWS API per eliminare il ruolo collegato al `AWSServiceRoleForAmazonEKSLocalOutpost` servizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

## Utilizzo dei ruoli per la dashboard Amazon EKS

La dashboard Amazon EKS utilizza questo ruolo collegato ai servizi per aggregare informazioni sulle risorse del cluster da più regioni e account. La dashboard si integra con AWS Organizations per leggere in modo sicuro le informazioni su più account dell'organizzazione.

Per ulteriori informazioni sulle dashboard Amazon EKS, consulta the section called "Pannello di controllo Amazon EKS".

Contesto

Amazon EKS utilizza ruoli AWS collegati al servizio Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo unico di ruolo IAM collegato direttamente ad Amazon EKS. Service-linked i ruoli sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS i servizi che funzionano con IAM e cerca i servizi con Sì nella colonna dei ruoli. Service-linked  Scegliere Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato al servizio per tale servizio.

Service-linked autorizzazioni di ruolo per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSDashboard`. Il ruolo consente ad Amazon EKS di generare e visualizzare la dashboard EKS, incluse informazioni aggregate sulle risorse del cluster. La policy `AmazonEKSDashboardServiceRolePolicy` allegata consente al ruolo di gestire le seguenti risorse: gruppi Auto Scaling, gruppi di sicurezza, modelli di avvio e profili dell'istanza IAM. Per ulteriori informazioni, consulta the section called "AWS policy gestita: AmazonEKSDashboardServiceRolePolicy".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSDashboard` considera attendibili i seguenti servizi:

- `dashboard.eks.amazonaws.com`

Per visualizzare la versione più recente del documento sulla policy JSON, consulta AmazonEKSDashboardServiceRolePolicyla AWS Managed Policy Reference Guide.

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta le autorizzazioni dei Service-linked ruoli nella Guida per l'utente IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando abiliti la dashboard nella AWS console, Amazon EKS crea il ruolo collegato al servizio per te.

Per ulteriori informazioni sull'abilitazione della dashboard EKS, consulta the section called "Configurare le organizzazioni".

> **Important**
>
> Questo ruolo collegato al servizio può apparire nell'account, se è stata completata un'operazione in un altro servizio che utilizza le caratteristiche supportate da questo ruolo.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSDashboard`. Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per ulteriori informazioni sulla disabilitazione della dashboard EKS, consulta the section called "Configurare le organizzazioni".

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la AWS CLI o l' AWS API per eliminare il ruolo collegato al `AWSServiceRoleForAmazonEKSDashboard` servizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

## Ruolo IAM di esecuzione del pod di Amazon EKS

Il ruolo di esecuzione di Amazon EKS Pod è necessario per eseguire Pods sull'infrastruttura AWS Fargate.

Quando il cluster crea Pods sull'infrastruttura AWS Fargate, i componenti in esecuzione sull'infrastruttura Fargate devono effettuare chiamate AWS APIs per conto dell'utente. In questo modo possono eseguire azioni come estrarre le immagini dei container da Amazon ECR o indirizzare i log ad altri AWS servizi. Il ruolo di esecuzione del pod Amazon EKS fornisce le autorizzazioni IAM per eseguire questa operazione.

Quando si crea un profilo Fargate, è necessario specificare un ruolo di esecuzione del pod per i componenti Amazon EKS che vengono eseguiti su infrastruttura Fargate utilizzando il profilo. Questo ruolo è aggiunto al controllo di accesso basato sul ruolo (RBAC) di Kubernetes del cluster per l'autorizzazione. Ciò consente al `kubelet` in esecuzione sull'infrastruttura Fargate di registrarsi con il cluster Amazon EKS in modo che possa essere visualizzato nel cluster come nodo.

> **Note**
>
> Il profilo Fargate deve avere un ruolo IAM diverso rispetto ai gruppi di EC2 nodi Amazon.

> **Important**
>
> I container in esecuzione nel pod Fargate non possono assumere le autorizzazioni IAM associate a un ruolo di esecuzione del pod. Per concedere ai contenitori del tuo Fargate Pod le autorizzazioni per accedere ad altri AWS servizi, devi utilizzare i ruoli IAM per gli account di servizio.

Prima di creare un profilo Fargate, devi creare un ruolo IAM con Amazon. EKSFargate PodExecutionRolePolicy

## Verificare la presenza di un ruolo di esecuzione del pod esistente configurato correttamente

Per verificare se l'account dispone già di un ruolo di esecuzione del pod Amazon EKS configurato correttamente, utilizzare la procedura indicata di seguito. Per prevenire il problema di sicurezza di un delegato confuso, è importante che il ruolo limiti l'accesso in base a `SourceArn`. È possibile modificare il ruolo di esecuzione secondo necessità per includere il supporto per i profili Fargate su altri cluster.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, cerca l'elenco dei ruoli per Amazon EKSFargate PodExecutionRole. Se il ruolo non esiste, consulta the section called "Creazione del ruolo di esecuzione del pod Amazon EKS" per crearlo. Se il ruolo è presente, selezionalo.
4. Nella EKSFargate PodExecutionRole pagina Amazon, procedi come segue:
   a. Seleziona Autorizzazioni.
   b. Assicurati che la policy gestita da EKSFargatePodExecutionRolePolicyAmazon Amazon sia associata al ruolo.
   c. Scegli Trust relationships (Relazioni di trust).
   d. Seleziona Edit trust policy (Modifica policy di attendibilità).

5. Nella pagina Edit trust policy (Modifica policy di attendibilità), verifica che la relazione di attendibilità contenga la policy seguente e una riga per i profili Fargate nel cluster. In tal caso, scegli Cancel (Annulla).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-east-1:111122223333:fargateprofile/my-
cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Se la policy corrisponde ma non contiene una riga che specifica i profili Fargate sul cluster, aggiungi la riga riportata di seguito nella parte superiore dell'oggetto `ArnLike`. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster, *`111122223333`* con l'ID dell'account e *`my-cluster`* con il nome del cluster.

```
"aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

Se la policy non corrisponde, copia la policy precedente nel modulo e scegli Update policy (Aggiorna policy). Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster. Se desideri utilizzare lo stesso ruolo in tutte le AWS regioni del tuo account, sostituiscilo *`region-code`* con*. Sostituisci *`111122223333`* con il tuo ID account e *`my-cluster`* con il nome del cluster. Se vuoi usare lo stesso ruolo per tutti i cluster dell'account, sostituisci *`my-cluster`* con *.

## Creazione del ruolo di esecuzione del pod Amazon EKS

Se non disponi già del ruolo di esecuzione di Amazon EKS Pod per il tuo cluster, puoi utilizzare Console di gestione AWS o la AWS CLI per crearlo.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Nel pannello di navigazione a sinistra, seleziona Ruoli.
c. Nella pagina Ruoli, seleziona Crea ruolo.
d. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
    i. Nella sezione Tipo di entità affidabile, scegli  AWS servizio.
    ii. Dall'elenco a discesa Casi d'uso per altri AWS servizi, scegli EKS.
    iii. Scegli EKS - Pod Fargate.
    iv. Scegli Next (Successivo).
e. Nella pagina Add permissions (Aggiungi autorizzazioni), scegli Next (Successivo).
f. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
    i. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSFargatePodExecutionRole`.
    ii. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
    iii. Scegli Crea ruolo.
g. Nella pagina Ruoli, cerca l'elenco dei ruoli per Amazon EKSFargate PodExecutionRole. Seleziona il ruolo.
h. Nella EKSFargate PodExecutionRole pagina Amazon, procedi come segue:
    i. Scegli Trust relationships (Relazioni di trust).
    ii. Seleziona Edit trust policy (Modifica policy di attendibilità).
i. Nella pagina Modifica policy di attendibilità, effettua le operazioni seguenti:
    i. Copia e incolla il contenuto seguente nel modello Modifica policy di attendibilità. Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster. Se desideri utilizzare lo stesso ruolo in tutte le AWS regioni del tuo account, sostituiscilo *`region-code`* con`*`. Sostituisci

*111122223333* con il tuo ID account e *my-cluster* con il nome del cluster. Se vuoi usare lo stesso ruolo per tutti i cluster dell'account, sostituisci *my-cluster* con *.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-
east-1:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ii. Scegli Aggiorna policy.

AWS CLI

a. Copia e incolla il contenuto seguente in un file denominato `pod-execution-role-trust-policy.json`. Sostituisci *region-code* con la AWS regione in cui si trova il cluster. Se desideri utilizzare lo stesso ruolo in tutte le AWS regioni del tuo account, sostituiscilo *region-code* con*. Sostituisci *111122223333* con il tuo ID account e *my-cluster* con il nome del cluster. Se vuoi usare lo stesso ruolo per tutti i cluster dell'account, sostituisci *my-cluster* con *.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-east-1:111122223333:fargateprofile/
my-cluster/*"
         }
```

```
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Creare un ruolo IAM di esecuzione del pod.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

c. Allega la policy IAM gestita da Amazon EKS richiesta al ruolo.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

## Ruolo IAM di Amazon EKS Connector

Puoi connettere i cluster Kubernetes per visualizzarli nel tuo. Console di gestione AWS Per collegarsi a un cluster Kubernetes, creare un ruolo IAM.

### Verificare la presenza di un ruolo del connettore EKS esistente

Per controllare se l'account dispone già di un ruolo di Amazon EKS Connector, utilizzare la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca l'elenco dei ruoli per `AmazonEKSConnectorAgentRole`. Se un ruolo che include `AmazonEKSConnectorAgentRole` non esiste, consultare the section called "Creazione del ruolo agente di Amazon EKS Connector" per crearlo. Se un ruolo che include `AmazonEKSConnectorAgentRole` esiste, seleziona il ruolo per visualizzare le policy allegate.
4. Selezionare Autorizzazioni.

5. Assicurati che la policy EKSConnector AgentPolicy gestita da Amazon sia associata al ruolo. Se la policy è collegata, il ruolo del connettore Amazon EKS è configurato correttamente.
6. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

## Creazione del ruolo agente di Amazon EKS Connector

Puoi usare Console di gestione AWS o AWS CloudFormation per creare il ruolo di agente del connettore.

AWS CLI

a. Creare un file denominato `eks-connector-agent-trust-policy.json` contenente il seguente JSON da utilizzare per il ruolo IAM.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Creare un file denominato `eks-connector-agent-policy.json` contenente il seguente JSON da utilizzare per il ruolo IAM.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws:eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

c. Crea il ruolo agente di Amazon EKS Connector utilizzando la policy di attendibilità e la policy creata negli elementi precedenti dell'elenco.

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

d. Allegare la policy al ruolo agente di Amazon EKS Connector.

```
aws iam put-role-policy \
```

```
    --role-name AmazonEKSConnectorAgentRole \
    --policy-name AmazonEKSConnectorAgentPolicy \
    --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

a. Salvate il seguente AWS CloudFormation modello in un file di testo sul vostro sistema locale.

> **Note**
>
> Questo modello crea anche il ruolo collegato al servizio che altrimenti verrebbe stato creato al momento della chiamata API `registerCluster`. Per informazioni dettagliate, vedi the section called "Ruolo del connettore del cluster".

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
```

```
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with AWS
 services.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

b. Apri la AWS CloudFormation console.

c. Scegliere Crea stack Con nuove risorse (standard).

d. In Specify template (Specifica modello), selezionare Upload a template file (Carica un file di modello) e Choose file (Scegli file).

e. Scegliere il file creato in precedenza, quindi selezionare Next (Successivo).

f. Per Stack name (Nome pila), immettere un nome per il ruolo, ad esempio `eksConnectorAgentRole`, quindi scegliere Next (Successivo).

g. Nella pagina Configure stack options (Configura opzioni pila), scegliere Next (Successivo).

h. Nella pagina Revisione, esaminare le informazioni, accettare che la pila può creare risorse IAM, quindi scegliere Crea pila.

## AWS policy gestite per Amazon Elastic Kubernetes Service

Una policy AWS gestita è una policy autonoma creata e amministrata da AWS. AWS le policy gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere autorizzazioni con privilegi minimi per i tuoi casi d'uso specifici perché sono disponibili per tutti i clienti. AWS Si consiglia pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i propri casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una policy AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui la policy è associata. AWS è più probabile che aggiorni una policy AWS gestita quando viene lanciato un nuovo AWS servizio o diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

## AWS politica gestita: Amazoneks_CNI_Policy

É possibile allegare la `AmazonEKS_CNI_Policy` alle entità IAM. Prima di creare un gruppo di nodi Amazon EC2, questa policy deve essere allegata al ruolo IAM del nodo, o a un ruolo IAM che è utilizzato in modo specifico dal plugin CNI di Amazon VPC per Kubernetes. In questo modo può eseguire operazioni per conto dell'utente. Si consiglia di allegare le policy a un ruolo utilizzato solo dal plugin. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC" e the section called "Configurare per IRSA".

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2:*NetworkInterface`** e **`ec2:*PrivateIpAddresses`** : consente al plugin CNI di Amazon VPC di eseguire azioni come il provisioning di interfacce di rete elastiche e indirizzi IP per i pod, così da fornire una rete per le applicazioni eseguite in Amazon EKS.
- Azioni di lettura **`ec2`**: consente al plugin CNI di Amazon VPC di eseguire azioni come descrivere istanze e sottoreti, così da visualizzare la quantità di indirizzi IP gratuiti nelle sottoreti Amazon VPC. Il CNI di VPC può utilizzare gli indirizzi IP gratuiti in ciascuna sottorete per scegliere le sottoreti con gli indirizzi IP più liberi da utilizzare durante la creazione di un'interfaccia di rete elastica.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazoneks_CNI_Policy nella Managed Policy Reference Guide. https://docs.aws.amazon.com/aws-managed-policy/latest/reference/AmazonEKS_CNI_Policy.html#AmazonEKS_CNI_Policy-json AWS

## AWS policy gestita: AmazonEKSClusterPolicy

È possibile allegare `AmazonEKSClusterPolicy` alle entità IAM. Prima di creare un cluster, è necessario disporre di un ruolo IAM del cluster con questa policy allegata. I cluster Kubernetes gestiti

da Amazon EKS effettuano chiamate ad altri AWS servizi per tuo conto. Ciò serve a gestire le risorse utilizzate con il servizio.

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`autoscaling`** : leggi e aggiorna la configurazione di un gruppo Auto Scaling. Queste autorizzazioni non sono utilizzate da Amazon EKS, ma rimangono nella policy per la compatibilità con le versioni precedenti.
- **`ec2`** : lavora con volumi e risorse di rete associati ai nodi Amazon EC2. Ciò è necessario affinché il piano di controllo (control-plane) Kubernetes possa unire le istanze a un cluster ed eseguire dinamicamente il provisioning e la gestione dei volumi Amazon EBS richiesti dai volumi persistenti di Kubernetes.
- **`ec2`** : elimina le interfacce di rete elastiche create da CNI di VPC. Ciò è necessario affinché EKS possa ripulire le interfacce di rete elastiche che rimangono se CNI di VPC si chiude inaspettatamente.
- **`elasticloadbalancing`** : lavora con Elastic Load Balancers e aggiunge nodi come destinazioni. Ciò è necessario affinché il piano di controllo Kubernetes possa eseguire dinamicamente il provisioning degli Elastic Load Balancers (load balancer Elastico) richiesti dai servizi Kubernetes.
- **`iam`** : crea un ruolo collegato ai servizi. Ciò è necessario affinché il piano di controllo (control-plane) Kubernetes possa eseguire dinamicamente il provisioning degli Elastic Load Balancer richiesti dai servizi Kubernetes.
- **`kms`**— Leggi una chiave da KMS. AWS Questa attività è necessaria affinché il piano di controllo (control-plane) Kubernetes supporti la crittografia dei segreti di Kubernetes archiviati in `etcd`.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta AmazonEKSClusterPolicy la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEKSDashboardConsoleReadOnly

È possibile collegare `AmazonEKSDashboardConsoleReadOnly` alle entità IAM.

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`eks`**- Read-only accesso ai dati del dashboard EKS, alle risorse e alle informazioni sulle versioni dei cluster. Ciò consente di visualizzare le EKS-related metriche e i dettagli di configurazione del cluster.
- **`organizations`**- Read-only accesso alle informazioni sulle AWS organizzazioni, tra cui:
  - visualizzazione dei dettagli dell'organizzazione e dell'accesso al servizio
  - elenco delle root organizzative, degli account e delle unità organizzative
  - Visualizzazione della struttura dell'organizzazione

Per visualizzare la versione più recente del documento sulla policy JSON, consulta AmazonEKSDashboardConsoleReadOnly la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEKSFargatePodExecutionRolePolicy

È possibile allegare `AmazonEKSFargatePodExecutionRolePolicy` alle entità IAM. Prima di poter creare un profilo Fargate, è necessario creare un ruolo di esecuzione del pod Fargate e allegare questa policy ad esso. Per ulteriori informazioni, consulta the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate" e the section called "Definisci i profili".

Questa politica concede al ruolo le autorizzazioni che forniscono l'accesso ad altre risorse di AWS servizio necessarie per eseguire Amazon EKS Pods su Fargate.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ecr`** : consente ai pod in esecuzione su Fargate di estrarre le immagini del container memorizzate in Amazon ECR.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference AmazonEKSFargatePodExecutionRolePolicy Guide. AWS

## AWS policy gestita: AmazonEKSConnectorServiceRolePolicy

Non è possibile allegare `AmazonEKSConnectorServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called "Ruolo del connettore del cluster".

Il ruolo consente ad Amazon EKS di collegarsi ai cluster Kubernetes. Le policy allegate consentono al ruolo di gestire le risorse necessarie per connettersi al cluster Kubernetes registrato.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`SSM Management`**— Crea, descrivi ed elimina le attivazioni SSM e annulla la registrazione delle istanze gestite. Ciò consente le operazioni di base di Systems Manager.
- **`Session Management`**— Avvia sessioni SSM specifiche per i cluster EKS ed esegui comandi non interattivi utilizzando il documento AmazonEKS.
- **`IAM Role Passing`**— Passa i ruoli IAM specificamente al servizio SSM, controllato da una condizione che limita i ruoli passati a. `ssm.amazonaws.com`
- **`EventBridge Rules`**— Crea EventBridge regole e obiettivi, ma solo se gestiti da. `eks-connector.amazonaws.com` Le regole sono specificamente limitate a AWS SSM come origine dell'evento.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta AmazonEKSConnectorServiceRolePolicy la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEKSForFargateServiceRolePolicy

Non è possibile allegare `AmazonEKSForFargateServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta `AWSServiceRoleForAmazonEKSForFargate`.

Questa policy concede ad Amazon EKS le autorizzazioni necessarie per eseguire le attività di Fargate. La policy viene utilizzato solo se si dispone di nodi Fargate.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`ec2`** : crea ed elimina interfacce di rete elastiche e descrive le interfacce di rete elastiche e le risorse. Questa operazione è necessaria affinché il servizio Fargate di Amazon EKS possa configurare la rete VPC necessaria per i pod Fargate.

Per visualizzare la versione più recente del documento sulla politica JSON, consulta AmazonEKSForFargateServiceRolePolicy la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEKSComputePolicy

È possibile collegare `AmazonEKSComputePolicy` alle entità IAM. È possibile allegare questa policy al ruolo IAM del cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy concede le autorizzazioni necessarie ad Amazon EKS per creare e gestire istanze EC2 per il cluster EKS e le autorizzazioni IAM necessarie per configurare EC2. Inoltre, questa policy concede ad Amazon EKS le autorizzazioni per creare il ruolo collegato al servizio Spot EC2 per conto dell'utente.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- Autorizzazioni**`ec2`** :
  - `ec2:CreateFleete` `ec2:RunInstances` - Consente di creare istanze EC2 e utilizzare risorse EC2 specifiche (immagini, gruppi di sicurezza, sottoreti, prenotazioni di capacità su richiesta, gruppi di posizionamento) per i nodi del cluster EKS.
  - `ec2:CreateLaunchTemplate`: consente di creare modelli di lancio EC2 per i nodi del cluster EKS.
  - La policy include anche condizioni per limitare l'utilizzo di queste autorizzazioni EC2 alle risorse contrassegnate con il nome del cluster EKS e altri tag pertinenti.
  - `ec2:CreateTags`: consente di aggiungere tag alle risorse EC2 create dalle azioni `CreateFleet`, `RunInstances` e `CreateLaunchTemplate`.
  - `ec2:DescribeCapacityReservations`- Consente di creare istanze EC2 da prenotazioni di capacità interrompibile
- Autorizzazioni **`iam`**:
  - `iam:AddRoleToInstanceProfile`: consente di aggiungere un ruolo IAM al profilo dell'istanza di calcolo EKS.
  - `iam:PassRole`: consente di passare i ruoli IAM necessari al servizio EC2.

Per visualizzare la versione più recente del documento sulla politica JSON, consulta la Managed Policy Reference AmazonEKSComputePolicy Guide. AWS

## AWS policy gestita: AmazonEKSNetworkingPolicy

È possibile collegare `AmazonEKSNetworkingPolicy` alle entità IAM. È possibile allegare questa policy al ruolo IAM del cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy è progettata per concedere le autorizzazioni necessarie ad Amazon EKS per creare e gestire interfacce di rete per il cluster EKS, consentendo al piano di controllo e ai nodi worker di comunicare e funzionare correttamente.

Dettagli delle autorizzazioni

Questa policy concede le seguenti autorizzazioni per consentire ad Amazon EKS di gestire le interfacce di rete per il cluster:

- Autorizzazioni dell'interfaccia di rete **`ec2`**
  - `ec2:CreateNetworkInterface`: consente la creazione di interfacce di rete EC2.
  - La policy include condizioni per limitare l'utilizzo di questa autorizzazione alle interfacce di rete contrassegnate con il nome del cluster EKS e il nome del nodo CNI di Kubernetes.
  - `ec2:CreateTags`: consente di aggiungere tag alle interfacce di rete create dall'azione `CreateNetworkInterface`.
- Autorizzazioni di gestione delle interfacce di rete **`ec2`**:
  - `ec2:AttachNetworkInterface`,`ec2:ModifyNetworkInterfaceAttribute`, `ec2:DetachNetworkInterface` - Consente di collegare, modificare gli attributi dell'interfaccia di rete e scollegare le interfacce di rete alle istanze EC2.
  - `ec2:UnassignPrivateIpAddresses`, `ec2:UnassignIpv6Addresses`, `ec2:AssignPrivateIpAddresses`, `ec2:AssignIpv6Addresses`: consente di gestire le assegnazioni degli indirizzi IP delle interfacce di rete.
  - Tali autorizzazioni sono limitate alle interfacce di rete contrassegnate con il nome del cluster EKS.

Per visualizzare la versione più recente del documento sulla policy JSON, AmazonEKSNetworkingPolicy consulta la Managed Policy Reference Guide. AWS

## AWS policy gestita: AmazonEKSBlockStoragePolicyV2

È possibile collegare `AmazonEKSBlockStoragePolicyV2` alle entità IAM. Puoi allegare questa policy al ruolo IAM del tuo cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy concede le autorizzazioni necessarie ad Amazon EKS per creare, gestire e mantenere volumi e snapshot EC2 per il cluster EKS, consentendo al piano di controllo e ai nodi worker di fornire e utilizzare l'archiviazione persistente come richiesto dai carichi di lavoro Kubernetes.

Dettagli delle autorizzazioni

Questa policy IAM concede le seguenti autorizzazioni per consentire ad Amazon EKS di gestire volumi e snapshot EC2:

- Autorizzazioni per la gestione dei volumi **`ec2`**:
  - `ec2:AttachVolume`, `ec2:DetachVolume`, `ec2:ModifyVolume`, `ec2:EnableFastSnapshotRestores`: consente di allegare, scollegare, modificare e abilitare ripristini rapidi di snapshot per i volumi EC2.
  - Queste autorizzazioni sono limitate ai volumi contrassegnati con il nome del cluster EKS.
  - `ec2:CreateTags`: consente di aggiungere tag ai volumi e agli snapshot di EC2 creati dalle azioni `CreateVolume` e `CreateSnapshot`.
- Autorizzazioni per la creazione di volumi **`ec2`**:
  - `ec2:CreateVolume`: consente la creazione di nuovi volumi EC2.
  - La policy include condizioni per limitare l'utilizzo di questa autorizzazione a volumi contrassegnati con il nome del cluster EKS e altri tag pertinenti.
  - `ec2:CreateSnapshot`: consente la creazione di nuovi snapshot di volumi EC2.
  - La policy include condizioni per limitare l'utilizzo di questa autorizzazione agli snapshot contrassegnati con il nome del cluster EKS e altri tag pertinenti.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta AmazonEKSBlockStoragePolicyV2 la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEKSLoadBalancingPolicy

È possibile collegare `AmazonEKSLoadBalancingPolicy` alle entità IAM. È possibile allegare questa policy al ruolo IAM del cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy IAM concede le autorizzazioni necessarie ad Amazon EKS per lavorare con vari AWS servizi per gestire Elastic Load Balancer (ELB) e le relative risorse.

Dettagli delle autorizzazioni

Le autorizzazioni principali concesse da questa policy sono:

- **`elasticloadbalancing`** : consente di creare, modificare e gestire Elastic Load Balancer e gruppi di destinazione. Ciò include le autorizzazioni per creare, aggiornare ed eliminare bilanciatori del carico, gruppi di destinazione, listener e regole.
- **`ec2`** : consente di creare e gestire gruppi di sicurezza, necessari al piano di controllo Kubernetes per unire le istanze a un cluster e gestire i volumi Amazon EBS. Consente inoltre di descrivere ed elencare risorse EC2 come istanze, VPC, sottoreti, gruppi di sicurezza e altre risorse di rete.
- **`iam`** : consente di creare un ruolo collegato al servizio per Elastic Load Balancing, necessario per il piano di controllo di Kubernetes per il provisioning dinamico di ELB.
- **`kms`**: consente di leggere una chiave da AWS KMS, necessaria affinché il piano di controllo di Kubernetes supporti la crittografia dei segreti Kubernetes archiviati in etcd.
- **`wafv2`**e **`shield`**: consente di elencare, associare e dissociare gli ACL Web e le protezioni Shield per Elastic Load Balancer. creating/deleting AWS
- **`cognito-idp`** , **`acm`** e **`elasticloadbalancing`** : concede le autorizzazioni per descrivere i client del pool di utenti, elencare e descrivere i certificati, e descrivere i gruppi di destinazione, necessari al piano di controllo di Kubernetes per gestire gli Elastic Load Balancer.

La policy include anche diversi controlli delle condizioni per garantire che le autorizzazioni rientrino nell'ambito dello specifico cluster EKS gestito, utilizzando il tag `eks:eks-cluster-name`.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference Guide. AmazonEKSLoadBalancingPolicy AWS

## AWS policy gestita: AmazonEKSMCPReadOnlyAccess

È possibile collegare `AmazonEKSMCPReadOnlyAccess` alle entità IAM. Questa policy fornisce l'accesso in sola lettura alle risorse Amazon EKS e ai AWS servizi correlati, consentendo al server Amazon EKS Model Context Protocol (MCP) di eseguire operazioni di osservabilità e risoluzione dei problemi senza apportare modifiche all'infrastruttura.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni che consentono ai committenti di completare le seguenti attività:

- **`eks`**— Consente ai principali di descrivere ed elencare i cluster EKS, i gruppi di nodi, i componenti aggiuntivi, le voci di accesso, gli approfondimenti e accedere all'API Kubernetes per operazioni di sola lettura.

- **`iam`**— Consente ai principali di recuperare informazioni sui ruoli IAM, sulle policy e sui relativi allegati per comprendere le autorizzazioni associate alle risorse EKS.
- **`ec2`**— Consente ai principali di descrivere VPC, sottoreti e tabelle di routing per comprendere la configurazione di rete dei cluster EKS.
- **`sts`**— Consente ai principali di recuperare le informazioni sull'identità del chiamante per scopi di autenticazione e autorizzazione.
- **`logs`**— Consente ai principali di avviare le query e recuperarne i risultati dai CloudWatch registri per la risoluzione dei problemi e il monitoraggio.
- **`cloudwatch`**— Consente ai responsabili di recuperare i dati metrici per il monitoraggio delle prestazioni dei cluster e dei carichi di lavoro.
- **`eks-mcp`**— Consente ai principali di richiamare le operazioni MCP e chiamare gli strumenti di sola lettura all'interno del server MCP Amazon EKS.

Per visualizzare la versione più recente del documento sulla policy JSON, AmazonEKSMCPReadOnlyAccess consulta la Managed Policy Reference Guide. AWS

## AWS policy gestita: AmazonEKSServicePolicy

È possibile collegare `AmazonEKSServicePolicy` alle entità IAM. I cluster creati prima del 16 aprile 2020 richiedono all'utente di creare un ruolo IAM e allegarvi questa policy. I cluster creati a partire dal 16 aprile 2020 non richiedono la creazione di un ruolo e non richiedono l'assegnazione di questa policy. Quando crei un cluster utilizzando un principal IAM che dispone dell'`iam:CreateServiceLinkedRole`autorizzazione, il ruolo AWSServiceRoleForAmazonEKS collegato al servizio viene creato automaticamente per te. Al ruolo collegato al servizio è associata la policy gestita. AmazonEKSServiceRolePolicy

Questa policy consente ad Amazon EKS di creare e gestire le risorse necessarie per gestire i cluster Amazon EKS.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`eks`** : aggiorna la versione Kubernetes del cluster dopo aver avviato un aggiornamento. Questa autorizzazione non è utilizzata da Amazon EKS, ma rimane nella policy per la compatibilità con le versioni precedenti.

- **`ec2`** : lavora con le interfacce di rete elastiche e altre risorse e tag di rete. Ciò è richiesto da Amazon EKS per configurare la rete che facilita la comunicazione tra i nodi e il piano di controllo Kubernetes. Leggi le informazioni sui gruppi di sicurezza. Aggiorna i tag sui gruppi di sicurezza.
- **`route53`** : associa un VPC a una zona ospitata. Ciò è richiesto da Amazon EKS per abilitare la rete di endpoint privati per il server API del cluster Kubernetes.
- **`logs`** : registra eventi. Ciò è necessario affinché Amazon EKS possa inviare i log del piano di controllo di Kubernetes a. CloudWatch
- **`iam`** : crea un ruolo collegato ai servizi. Questa operazione è necessaria affinché Amazon EKS possa creare il ruolo collegato al servizio the section called "Service-linked autorizzazioni di ruolo per Amazon EKS" per conto dell'utente.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la AWS Managed Policy AmazonEKSServicePolicy Reference Guide.

## AWS policy gestita: AmazonEKSServiceRolePolicy

Non è possibile allegare `AmazonEKSServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called "Service-linked autorizzazioni di ruolo per Amazon EKS". Quando crei un cluster utilizzando un principal IAM che dispone dell'`iam:CreateServiceLinkedRole`autorizzazione, il ruolo AWSServiceRoleForAmazonEKS collegato al servizio viene creato automaticamente per te e questa policy è associata ad esso.

Questa policy consente al ruolo collegato al servizio di chiamare i AWS servizi per tuo conto.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`ec2`**— Crea e descrivi le interfacce di rete elastiche, le istanze Amazon EC2 e i relativi stati, il gruppo di sicurezza del cluster e il VPC necessari per creare un cluster. Per ulteriori informazioni, consulta the section called "Requisiti relativi al gruppo di sicurezza". Leggi le informazioni sui gruppi di sicurezza. Aggiorna i tag sui gruppi di sicurezza. Leggi le informazioni su Capacity Reservations. On-Demand Leggi le informazioni sui gruppi di collocamento. Leggi la configurazione del VPC, incluse le tabelle di routing e gli ACL di rete, per rilevare i problemi di configurazione come parte degli approfondimenti sui cluster.

- Modalità automatica **`ec2`**: termina le istanze EC2 create dalla modalità automatica EKS. Per ulteriori informazioni, consulta Modalità automatica EKS.
- **`iam`**— Elenca tutte le policy gestite associate a un ruolo IAM. Ciò è necessario affinché Amazon EKS possa pubblicare e convalidare tutte le politiche gestite e le autorizzazioni necessarie per la creazione di un cluster.
- Associa un VPC a una zona ospitata. Ciò è richiesto da Amazon EKS per abilitare la rete di endpoint privati per il server API del cluster Kubernetes.
- Evento di registro: è necessario affinché Amazon EKS possa inviare i log del piano di controllo di Kubernetes a. CloudWatch
- Inserisci una metrica: è necessaria affinché Amazon EKS possa inviare i log del piano di controllo di Kubernetes a. CloudWatch
- **`eks`** : gestisci le voci e le politiche di accesso ai cluster, permettendo un controllo granulare su chi può accedere alle risorse EKS e quali azioni può eseguire. Ciò include l'associazione di policy di accesso standard per operazioni di elaborazione, rete, bilanciamento del carico e archiviazione.
- **`elasticloadbalancing`** : crea, gestisci ed elimina i bilanciatori del carico e i relativi componenti (listener, gruppi di destinazione, certificati) associati ai cluster EKS. Visualizza gli attributi e lo stato di integrità del bilanciatore del carico.
- **`events`**- Crea e gestisci EventBridge regole per il monitoraggio degli eventi EC2 e AWS Health relativi ai cluster EKS, consentendo risposte automatiche alle modifiche dell'infrastruttura e agli avvisi sullo stato di salute.
- **`iam`**- Gestisci i profili delle istanze EC2 con il prefisso «eks», inclusi l'elenco, la creazione, l'eliminazione e l'associazione dei ruoli, necessari per la gestione dei nodi EKS. Consente di descrivere qualsiasi profilo di istanza per consentire agli utenti di definire profili di istanza personalizzati da utilizzare per i propri nodi di lavoro.
- **`pricing`**& **`shield`** - Accedi alle informazioni AWS sui prezzi e allo stato della protezione Shield, consentendo la gestione dei costi e le funzionalità di sicurezza avanzate per le risorse EKS.
- Pulizia delle risorse: elimina in modo sicuro EKS-tagged le risorse tra cui volumi, istantanee, modelli di avvio e interfacce di rete durante le operazioni di pulizia dei cluster.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference AmazonEKSServiceRolePolicy Guide. AWS

## AWS policy gestita: AmazonEKSVPCResourceController

È possibile allegare la policy `AmazonEKSVPCResourceController` alle identità IAM. Se utilizzi gruppi di sicurezza per pod, dovrai allegare questa policy al ruolo IAM del cluster Amazon EKS perché possa eseguire operazioni per tuo conto.

Questa policy concede le autorizzazioni del ruolo cluster per gestire le interfacce di rete elastiche e gli indirizzi IP per i nodi.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2`** : gestisci le interfacce di rete elastiche e gli indirizzi IP per supportare i gruppi di sicurezza dei pod e i nodi Windows.

Per visualizzare la versione più recente del documento sulla politica JSON, consulta AmazonEKSVPCResourceController la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEKSWorkerNodePolicy

É possibile allegare la `AmazonEKSWorkerNodePolicy` alle entità IAM. È necessario allegare questa policy a un Ruolo IAM del nodo che va specificato quando si creano nodi Amazon EC2 che consentono ad Amazon EKS di eseguire operazioni per conto dell'utente. Se si crea un gruppo di nodi utilizzando `eksctl`, si crea il ruolo IAM del nodo e si allega automaticamente questa policy al ruolo.

Questa policy concede ai nodi Amazon EKS Amazon EC2 le autorizzazioni per connettersi ai cluster Amazon EKS.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2`** : leggi le informazioni sul volume dell'istanza e sulla rete. Ciò è necessario affinché i nodi Kubernetes possano descrivere le informazioni sulle risorse Amazon EC2 necessarie per l'unione del nodo al cluster Amazon EKS.

- **`eks`** : descrivi in modo facoltativo il cluster come parte del bootstrap del nodo.
- **`eks-auth:AssumeRoleForPodIdentity`** : consenti il recupero delle credenziali per i carichi di lavoro EKS sul nodo. Ciò è necessario per il funzionamento di EKS Pod Identity.

Per visualizzare la versione più recente del documento sulla politica JSON, consulta AmazonEKSWorkerNodePolicy la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEKSWorkerNodeMinimalPolicy

Puoi collegarlo AmazonEKSWorkerNodeMinimalPolicy alle tue entità IAM. È necessario allegare questa policy a un ruolo IAM del nodo che deve essere specificato quando si creano nodi Amazon EC2 che consentono ad Amazon EKS di eseguire operazioni per conto dell'utente.

Questa policy concede ai nodi Amazon EKS Amazon EC2 le autorizzazioni per connettersi ai cluster Amazon EKS. Questa policy ha meno autorizzazioni rispetto a AmazonEKSWorkerNodePolicy.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- `eks-auth:AssumeRoleForPodIdentity`: consenti il recupero delle credenziali per i carichi di lavoro EKS sul nodo. Ciò è necessario per il funzionamento di EKS Pod Identity.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta AmazonEKSWorkerNodeMinimalPolicy la AWS Managed Policy Reference Guide.

## AWS policy gestita: AWSServiceRoleForAmazonEKSNodegroup

Non è possibile allegare `AWSServiceRoleForAmazonEKSNodegroup` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called "Service-linked autorizzazioni di ruolo per Amazon EKS".

Questa policy concede l'autorizzazione ai ruoli `AWSServiceRoleForAmazonEKSNodegroup` che consentono di creare e gestire gruppi di nodi Amazon EC2 nell'account.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2`** : lavora con gruppi di sicurezza, tag, prenotazioni della capacità e modelli di avvio. Ciò è necessario per i gruppi di nodi gestiti da Amazon EKS per abilitare la configurazione dell'accesso remoto e descrivere le prenotazioni della capacità che possono essere utilizzate nei gruppi di nodi gestiti. Inoltre, i gruppi di nodi gestiti da Amazon EKS creano un modello di avvio per conto dell'utente. Questo consente di configurare il gruppo Amazon EC2 Auto Scaling che supporta ogni gruppo di nodi gestito.
- **`iam`** : crea un ruolo collegato ai servizi e passa un ruolo. Ciò è richiesto dai gruppi di nodi gestiti da Amazon EKS per gestire i profili di istanza per il ruolo passato durante la creazione di un gruppo di nodi gestito. Questo profilo dell'istanza viene utilizzato dalle istanze Amazon EC2 avviate come parte di un gruppo di nodi gestito. Amazon EKS deve creare ruoli collegati al servizio per altri servizi come, ad esempio i gruppi di Amazon EC2 Auto Scaling. Queste autorizzazioni vengono utilizzate nella creazione di un gruppo di nodi gestito.
- **`autoscaling`** : lavora con gruppi Auto Scaling di sicurezza. Questo è richiesto dai gruppi di nodi gestiti da Amazon EKS per gestire il gruppo di Amazon EC2 Auto Scaling che supporta ciascun gruppo di nodi gestito. Viene anche utilizzato per supportare funzionalità come l'eliminazione dei pod quando i nodi vengono terminati o riciclati durante gli aggiornamenti dei gruppi di nodi e la gestione dei warm pool configurati su gruppi di nodi gestiti.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference AWSServiceRoleForAmazonEKSNodegroup Guide. AWS

## AWS policy gestita: AmazonEKSDashboardServiceRolePolicy

Non è possibile allegare `AmazonEKSDashboardServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called "Service-linked autorizzazioni di ruolo per Amazon EKS".

Questa politica concede al `AWSServiceRoleForAmazonEKSDashboard` ruolo le autorizzazioni che gli consentono di visualizzare informazioni sulla struttura e sugli account dell' AWS organizzazione.

Dettagli delle autorizzazioni

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`organizations`**— Visualizza informazioni sulla struttura e sugli account della tua AWS organizzazione. Ciò include le autorizzazioni per elencare gli account dell'organizzazione, visualizzare le unità organizzative e le root, elencare gli amministratori delegati, visualizzare i servizi che hanno accesso all'organizzazione e recuperare informazioni dettagliate sull'organizzazione e sugli account.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta AmazonEKSDashboardServiceRolePolicy la AWS Managed Policy Reference Guide.

## AWS policy gestita: AmazonEBSCSIDriverPolicy

> **Important**
>
> Sono disponibili due nuove policy IAM AWS gestite che forniscono un ambito di autorizzazione più restrittivo: AmazonEBSCSIDriverPolicyV2 per l'ambito basato su tag o per l'isolamento con ambito di cluster. AmazonEBSCSIDriverEKSClusterScopedPolicy Se sei interessato alla migrazione, consulta la sezione EBS CSI Driver policy migration. https://github.com/kubernetes-sigs/aws-ebs-csi-driver/issues/2918

La `AmazonEBSCSIDriverPolicy` policy consente al driver Amazon EBS Container Storage Interface (CSI) di creare, modificare, copiare, allegare, scollegare ed eliminare volumi per tuo conto. Ciò include la modifica dei tag sui volumi esistenti e l'abilitazione del ripristino rapido degli snapshot (FSR) sui volumi EBS. Inoltre, concede al driver EBS CSI le autorizzazioni per creare, bloccare, ripristinare ed eliminare istantanee e per elencare istanze, volumi e istantanee.

Per visualizzare l'ultima versione del documento sulla policy JSON, consulta la Managed Policy Reference Guide. AmazonEBSCSIDriverServiceRolePolicy AWS

## AWS policy gestita: AmazonEBSCSIDriverPolicyV2

La `AmazonEBSCSIDriverPolicyV2` politica è un'alternativa più restrittiva a`AmazonEBSCSIDriverPolicy`. Limita il driver CSI di Amazon EBS alla sola gestione dei volumi EBS e delle istantanee contrassegnati con la chiave impostata su. `ebs.csi.aws.com/cluster` `true` I volumi forniti dal plug-in di volume Kubernetes integrato nell'albero (CSI-migrated volumi) sono supportati anche tramite il tag resource. `kubernetes.io/created-for/pvc/name`

In caso di migrazione da`AmazonEBSCSIDriverPolicy`, consulta la sezione EBS CSI Driver policy migration.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la AWS Managed Policy AmazonEBSCSIDriverPolicyV2 Reference Guide.

## AWS policy gestita: AmazonEBSCSIDriverEKSClusterScopedPolicy

La `AmazonEBSCSIDriverEKSClusterScopedPolicy` policy limita il driver CSI di Amazon EBS alla gestione solo dei volumi EBS e delle istantanee che appartengono a un cluster EKS specifico. Richiede che il tag della risorsa `ebs.csi.aws.com/cluster-name` corrisponda al `eks-cluster-name` tag sul principale IAM, impedendo l'accesso tra cluster quando più cluster condividono lo stesso account. AWS Le operazioni di collegamento e scollegamento sulle istanze sono limitate alle istanze contrassegnate con il `eks:cluster-name` tag (impostato automaticamente da EKS sui gruppi di nodi gestiti), il `eks:eks-cluster-name` tag (impostato dai controller che seguono lo standard di etichettatura dei controller EKS, inclusi Karpenter open source e Amazon EKS Auto Mode) o il `ebs.csi.aws.com/cluster-name` tag (per le istanze con tag manuale).

In caso di migrazione da, consulta la sezione EBS CSI Driver policy `AmazonEBSCSIDriverPolicy` migration. https://github.com/kubernetes-sigs/aws-ebs-csi-driver/issues/2918

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la AWS Managed Policy AmazonEBSCSIDriverEKSClusterScopedPolicy Reference Guide.

## AWS policy gestita: AmazonEFSCSIDriverPolicy

La`AmazonEFSCSIDriverPolicy`la policy consente all'Amazon EFS Container Storage Interface (CSI) di creare ed eliminare punti di accesso per tuo conto. Inoltre, concede al driver Amazon EFS CSI le autorizzazioni per elencare i punti di accesso, i file system, le destinazioni di montaggio e le zone di disponibilità di Amazon EC2.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference AmazonEFSCSIDriverPolicy Guide. AWS

## AWS policy gestita: AmazonS3FilesCSIDriverPolicy

La `AmazonS3FilesCSIDriverPolicy` policy consente ad Amazon EFS Container Storage Interface (CSI) di creare ed eliminare punti di accesso Amazon S3 Files per tuo conto. Inoltre, concede al driver Amazon EFS CSI le autorizzazioni per elencare i punti di accesso e i file system di Amazon S3 Files.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference AmazonS3FilesCSIDriverPolicy Guide. AWS

## AWS policy gestita: AmazonEKSLocalOutpostClusterPolicy

È possibile collegare questa policy alle entità IAM. Prima di creare un cluster, è necessario allegare questa policy a un ruolo cluster. I cluster Kubernetes gestiti da Amazon EKS effettuano chiamate ad altri AWS servizi per tuo conto. Ciò serve a gestire le risorse utilizzate con il servizio.

La `AmazonEKSLocalOutpostClusterPolicy` include le autorizzazioni seguenti:

- Azioni di lettura **`ec2`**: consente alle istanze del piano di controllo di descrivere le proprietà della zona di disponibilità, della tabella di routing, dell'istanza e dell'interfaccia di rete. Autorizzazioni necessarie affinché le istanze Amazon EC2 si uniscano correttamente al cluster come istanze del piano di controllo.
- **`ssm`** : consente la connessione di Systems Manager di Amazon EC2 all'istanza del piano di controllo, utilizzata da Amazon EKS per comunicare e gestire il cluster locale nel tuo account.
- **`logs`**— Consente alle istanze di inviare i log ad Amazon. CloudWatch
- **`secretsmanager`**— Consente alle istanze di ottenere ed eliminare i dati di bootstrap per le istanze del piano di controllo in modo sicuro da Secrets Manager. AWS
- **`ecr`** : consente a pod e container in esecuzione sulle istanze del piano di controllo di estrarre le immagini di container memorizzate in Amazon Elastic Container Registry.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference AmazonEKSLocalOutpostClusterPolicy Guide. AWS

## AWS policy gestita: AmazonEKSLocalOutpostServiceRolePolicy

Non è possibile allegare questa policy alle entità IAM. Quando si crea un cluster tramite un principale IAM che dispone dell'autorizzazione `iam:CreateServiceLinkedRole`, Amazon EKS crea automaticamente il ruolo collegato al servizio AWSServiceRoleForAmazonEKSLocalOutpost per conto tuo e questa policy è collegata ad esso. Questa policy consente al ruolo collegato al servizio di chiamare AWS i servizi per conto dell'utente per i cluster locali.

La `AmazonEKSLocalOutpostServiceRolePolicy` include le autorizzazioni seguenti:

- **`ec2`** : consente ad Amazon EKS di utilizzare le risorse di sicurezza, rete e altro tipo per avviare e gestire correttamente le istanze del piano di controllo nel tuo account.
- **`ssm`**, **`ssmmessages`** : consente la connessione di Amazon EC2 Systems Manager alle istanze del piano di controllo, utilizzata da Amazon EKS per comunicare e gestire il cluster locale nel tuo account.

- **`iam`** : consente ad Amazon EKS di gestire il profilo dell'istanza associato alle istanze del piano di controllo.
- **`secretsmanager`**- Consente ad Amazon EKS di inserire i dati di bootstrap per le istanze del piano di controllo in AWS Secrets Manager in modo che possano essere referenziati in modo sicuro durante il bootstrap dell'istanza.
- **`outposts`** : consente ad Amazon EKS di ottenere le informazioni sull'Outpost dal tuo account per avviare correttamente un cluster locale in un Outpost.

Per visualizzare la versione più recente del documento sulla policy JSON, AmazonEKSLocalOutpostServiceRolePolicy consulta la Managed Policy Reference Guide. AWS

## Amazon EKS si aggiorna a AWS policy gestite

Visualizza i dettagli sugli aggiornamenti delle policy AWS gestite per Amazon EKS da quando questo servizio ha iniziato a tracciare queste modifiche.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/security/iam-reference/security-iam-awsmanpol.adoc.atom
```

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Autorizzazioni aggiornate per. the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy" | Sono state aggiunte `gateway.eks.amazonaws.com.alb/stack` le chiavi `gateway.eks.amazonaws.com.alb/resource` , `gateway.eks.amazonaws.com.nlb/stack` , e `gateway.eks.amazonaws.com.nlb/resource` tag alla lista consentita per la codifica delle risorse in. `AmazonEKSLoadBalancingPolic` | 2 settembre 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| | y Ciò consente ad Amazon EKS Auto Mode Controller di gestire AWS le risorse per i carichi di lavoro Kubernetes Gateway API (ALB e NLB). | |
| Autorizzazione aggiunta a the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy". | È stata aggiunta `wafv2:ListWebACLs` l'autorizzazione in. `AmazonEKSLoadBalancingPolicy` Ciò consente ad Amazon EKS Auto Mode Controller di elencare gli ACL Web AWS WAF durante la riconciliazione delle associazioni Application Load Balancer. | 2 settembre 2026 |
| Autorizzazione aggiunta a AmazonEKSServiceRolePolicy. | Aggiunto `AmazonEKSPodIdentityPolicy` all'elenco delle politiche di accesso che il ruolo collegato al servizio può associare e dissociare dalle voci di accesso al cluster. Con questa aggiunta, Amazon EKS può gestire le credenziali di identità dei pod per i carichi di lavoro in esecuzione su cluster EKS. | 26 giugno 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Autorizzazioni aggiornate per. the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy" | Aggiunte `elasticloadbalancing:SetRulePriorities` , `ec2:DescribeIpamPools` autorizzazioni in. `AmazonEKSLoadBalancingPolicy` Ciò consente ad Amazon EKS Auto Mode Controller di impostare le priorità delle regole di listener specificate per ALB e descrivere rispettivamente i pool IPAM. | 15 giugno 2026 |
| Aggiunte autorizzazioni a AmazonEBSCSIDriverEKSClusterScopedPolicy. | Autorizzata `ec2:AttachVolume` e `ec2:DetachVolume` sulle istanze contrassegnate con il `eks:eks-cluster-name` tag, aggiungendo la compatibilità con i controller che seguono lo standard di etichettatura dei controller EKS, tra cui Karpenter open source e Amazon EKS Auto Mode. | 28 maggio 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Autorizzazione aggiunta a the section called "AWS policy gestita: AmazonEKS ComputePolicy". | Sono state aggiunte le autorizzazioni relative alle risorse per includere l'azione`ec2:DescribeCapacityReservations` . Ciò consente ad Amazon EKS Auto Mode di avviare istanze da Interruptible Capacity Reservations nel tuo account. | 18 maggio 2026 |
| Introdotto the section called "AWS policy gestita: AmazonEKSBlockStorage PolicyV2". | AWS ha introdotto il. `AmazonEKSBlockStorage PolicyV2` | 12 maggio 2026 |
| Aggiunte autorizzazioni a AmazonEBSCSIDriverPolicy. | È stata aggiunta una nuova azione API `ec2:DescribeInstanceTypes` per consentire al driver EBS CSI di recuperare dinamicamente le informazioni sul volume in fase di esecuzione. limit/card | 12 maggio 2026 |
| Aggiunta l'autorizzazione a the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy" | È stata aggiunta `elasticloadbalancing:DescribeLoadBalancers` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per consentire ad Amazon EKS Auto Mode Controller di descrivere i sistemi di bilanciamento del carico. | 27 aprile 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunta l'autorizzazione a the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy" | È stata aggiunta `elasticloadbalancing:DescribeCapacityReservation` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per consentire ad Amazon EKS Auto Mode Controller di descrivere le unità LoadBalancer di capacità | 20 aprile 2026 |
| Aggiunta l'autorizzazione a the section called "AWS policy gestita: AmazonEKS ServiceRolePolicy" | Aggiunti `ec2:DescribeInstanceStatus` e `iam:ListInstanceProfiles` autorizzazioni in. `AmazonEKSServiceRolePolicy` Queste autorizzazioni consentono ad Amazon EKS Auto Mode di rilevare gli eventi di integrità di EC2 per le istanze gestite e di pulire i profili delle istanze gestite | 20 aprile 2026 |
| Autorizzazione aggiunta a the section called "AWS policy gestita: AmazonEKS ComputePolicy". | Sono state aggiornate le autorizzazioni relative alle risorse `ec2:RunInstances` e le `ec2:CreateFleet` azioni per includere i gruppi di collocamento. `arn:aws:ec2:*:*:placement-group/*` Ciò consente ad Amazon EKS Auto Mode di avviare istanze nei gruppi di collocamento EC2 del tuo account. | 17 aprile 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Introdotto AmazonEBSCSIDriverPolicyV2 e. AmazonEBSCSIDriverEKSClusterScopedPolicy | Introdotto `AmazonEBSCSIDriverPolicyV2` e `AmazonEBSCSIDriverEKSClusterScopedPolicy` come alternative più restrittive a`AmazonEBSCSIDriverPolicy` . `AmazonEBSCSIDriverPolicyV2` assegna il driver EBS CSI ai volumi e alle istantanee contrassegnati con set to. `ebs.csi.aws.com/cluster` `true` `AmazonEBSCSIDriverEKSClusterScopedPolicy` assegna il driver ai volumi e alle istantanee appartenenti a uno specifico cluster EKS utilizzando il tag. `ebs.csi.aws.com/cluster-name` | 16 aprile 2026 |
| Autorizzazione aggiunta a AmazonEKSServiceRolePolicy. | È stata aggiunta `ec2:DescribePlacementGroups` l'autorizzazione per consentire ad Amazon EKS Auto Mode di avviare istanze nei gruppi di collocamento EC2 del tuo account. | 15 aprile 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Autorizzazioni aggiornate per. the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy" | Aggiornato`shield:CreateProtection` , `shield:DeleteProtection` autorizzazione in`AmazonEKSLoadBalancingPolicy` . Ciò consente ad Amazon EKS Auto Mode Controller di aggiungere una protezione Shield. | 14 aprile 2026 |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy". | `elasticloadbalancing:RegisterTargets` Autorizzazione aggiornata a `AmazonEKSLoadBalancingPolicy` per aggiungere il supporto per targetgroupbindings multi-cluster e personalizzati | 20 marzo 2026 |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy". | È stata aggiunta `elasticloadbalancing:ModifyIpPools` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per aggiungere il supporto per l'impostazione del pool IPAM per ALB | 20 marzo 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Autorizzazione aggiunta a the section called "AWS policy gestita: AmazonEKSNetworkingPolicy". | È stata aggiunta `ec2:ModifyNetworkInterfaceAttribute` l'autorizzazione in. `AmazonEKSNetworkingPolicy` Ciò consente ad Amazon EKS Auto Mode Controller di modificare gli attributi dell'interfaccia di rete relativi alle istanze EC2 | 3 febbraio 2026 |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AWSServiceRoleForAmazonEKSNodegroup". | Aggiunto`autoscaling:PutWarmPool` ,`autoscaling:DeleteWarmPool` , e `autoscaling:DescribeWarmPool` autorizzazioni a. `AWSServiceRoleForAmazonEKSNodegroup` Ciò consente ad Amazon EKS Managed Nodegroups di gestire le risorse del warm pool ASG sottostanti durante tutto il ciclo di vita del gruppo di nodi. | 17 febbraio 2026 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Autorizzazione aggiunta a the section called "AWS policy gestita: AmazonEKS ServiceRolePolicy". | È stato rimosso il requisito del prefisso «eks» nel nome del profilo dell'istanza di destinazione per l'`iam:GetInstanceProfile` autorizzazione all'accesso. `AmazonEKSServiceRolePolicy` Ciò consente ad Amazon EKS Auto Mode di convalidare e utilizzare profili di istanza personalizzati NodeClasses senza richiedere il prefisso di denominazione «eks». | 2 febbraio 2026 |
| Aggiunte autorizzazioni a AmazonEBSCSIDriverPolicy. | È stata aggiunta `ec2:LockSnapshot` l'autorizzazione per consentire al driver EBS CSI di bloccare direttamente le istantanee EBS. | 15 gennaio 2026 |
| Introdotto the section called "AWS policy gestita: AmazonEKSMCPReadOnlyAccess". | Amazon EKS ha introdotto una nuova policy gestita `AmazonEKSMCPReadOnlyAccess` per abilitare strumenti di sola lettura nel server MCP Amazon EKS per l'osservabilità e la risoluzione dei problemi. | 21 novembre 2025 |
| Aggiunte autorizzazioni a AmazonEBSCSIDriverPolicy. | È stata aggiunta `ec2:CopyVolumes` l'autorizzazione per consentire al driver EBS CSI di copiare direttamente i volumi EBS. | 17 novembre 2025 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Autorizzazione aggiunta a the section called "AWS policy gestita: AmazonEKS ServiceRolePolicy". | Aggiunti `ec2:DescribeRouteTables` e `ec2:DescribeNetworkAcls` autorizzazioni a. `AmazonEKSServiceRolePolicy` Ciò consente ad Amazon EKS di rilevare problemi di configurazione con le tabelle di routing VPC e gli ACL di rete per i nodi ibridi come parte delle informazioni sui cluster. | 22 ottobre 2025 |
| Aggiunta l'autorizzazione a AWSServiceRoleForAmazonEKSConnector | È stata aggiunta `ssmmessages:OpenDataChannel` l'autorizzazione a `AmazonEKSConnectorServiceRolePolicy` | 15 ottobre 2025 |
| Aggiunta l'autorizzazione a the section called "AWS policy gestita: AmazonEKS ServiceRolePolicy" | Questo ruolo può allegare una nuova policy di accesso `AmazonEKSEventPolicy` . Autorizzazioni limitate per `ec2:DeleteLaunchTemplate` e `ec2:TerminateInstances` . | 26 agosto 2025 |
| Aggiunta l'autorizzazione a the section called "AWS policy gestita: AmazonEKS LocalOutpostServiceRolePolicy" | Aggiunta l'autorizzazione `ssmmessages:OpenDataChannel` a `AmazonEKSLocalOutpostServiceRolePolicy` . | 26 giugno 2025 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunta l'autorizzazione a the section called "AWS policy gestita: AmazonEKS ComputePolicy". | Autorizzazioni aggiornate per le risorse `ec2:RunInstances` e `ec2:CreateFleet` azioni che includono le prenotazioni della capacità `arn:aws:ec2:*:*:capacity-reservation/*` . Ciò consente ad Amazon EKS Auto Mode di avviare istanze utilizzando le prenotazioni di On-Demand capacità EC2 nel tuo account. `iam:CreateServiceLinkedRole` aggiunto per consentire alla modalità automatica di Amazon EKS di creare il ruolo `AWSServiceRoleForEC2Spot` collegato al servizio Spot EC2 per tuo conto. | 20 giugno 2025 |
| Autorizzazione aggiunta a AmazonEKSServiceRolePolicy. | È stata aggiunta `ec2:DescribeCapacityReservations` l'autorizzazione per consentire ad Amazon EKS Auto Mode di avviare le istanze utilizzando le prenotazioni di On-Demand capacità EC2 nel tuo account. | 20 giugno 2025 |
| Introdotto the section called "AWS policy gestita: AmazonEKSDashboardConsoleReadOnly". | È stata introdotta una nuova policy `AmazonEKSDashboardConsoleReadOnly` . | 19 giugno 2025 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Introdotto the section called "AWS policy gestita: AmazonEKSDashboard ServiceRolePolicy". | È stata introdotta una nuova policy `AmazonEKS DashboardServiceRo lePolicy` . | 21 maggio 2025 |
| Aggiunte autorizzazioni a AmazonEKSClusterPolicy. | È stata aggiunta l'autoriz zazione `ec2:Delet eNetworkInterfaces` per consentire ad Amazon EKS di eliminare le interfacc e di rete elastiche che sono lasciate indietro se il CNI di VPC si chiude inaspetta tamente. | 16 aprile 2025 |
| Autorizzazione aggiunta a AmazonEKSServiceRo lePolicy. | Sono state aggiunte `ec2:RevokeSecurity GroupEgress` le `ec2:AuthorizeSecur ityGroupEgress` autorizzazioni per consentir e ai AI/ML clienti EKS di aggiungere regole Security Group Egress all'EKS Cluster SG predefinito compatibili con EFA, come parte della versione EKS 1.33. | 14 aprile 2025 |
| Aggiunte autorizzazioni a AmazonEKSServiceRo lePolicy. | Aggiunta l'autorizzazione per terminare le istanze EC2 create dalla modalità automatica di EKS. | 28 febbraio 2025 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a AmazonEBSCSIDriverPolicy. | È stata aggiunta una nuova dichiarazione che autorizza il driver CSI EBS a ripristin are tutti gli snapshot. Ciò era precedentemente consentit o dalla policy esistente, ma è necessaria una nuova dichiarazione esplicita per via di una modifica nella gestione di IAM per `CreateVolume` .<br><br>È stata aggiunta la possibili tà per il driver CSI EBS di modificare i tag sui volumi esistenti. Il driver EBS CSI può modificare i tag dei volumi esistenti tramite un parametro in Kubernetes. VolumeAtt ributesClasses<br><br>È stata aggiunta la possibili tà per il driver CSI EBS di abilitare il ripristino rapido degli snapshot (FSR) sui volumi EBS. Il driver CSI EBS può abilitare FSR su nuovi volumi tramite parametri nelle classi di archiviazione Kubernetes. | 13 gennaio 2025 |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AmazonEKS LoadBalancingPolicy". | Aggiornato `AmazonEKS LoadBalancingPolic y` per consentire l'elenco e la descrizione delle risorse di rete e degli indirizzi IP. | 26 dicembre 2024 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AWSServiceRoleForAmazonEKSNodegroup". | Aggiornato `AWSServiceRoleForAmazonEKSNodegroup` per compatibilità con le regioni della Cina. | 22 novembre 2024 |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AmazonEKSLocalOutpostClusterPolicy" | È stata aggiunta l'`ec2:DescribeAvailabilityZones` autorizzazione `AmazonEKSLocalOutpostClusterPolicy` in modo che il AWS Cloud Controller Manager sul piano di controllo del cluster possa identificare la zona di disponibilità in cui si trova ciascun nodo. | 21 novembre 2024 |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AWSServiceRoleForAmazonEKSNodegroup". | Policy `AWSServiceRoleForAmazonEKSNodegroup` aggiornata per consentire a `ec2:RebootInstances` le istanze create da gruppi di nodi gestiti di Amazon EKS. Limitazione delle autorizzazioni `ec2:CreateTags` per le risorse Amazon EC2. | 20 novembre 2024 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a the section called "AWS policy gestita: AmazonEKS ServiceRolePolicy". | Politica AWS gestita aggiornata di EKS `AmazonEKSServiceRolePolicy`. Aggiunte le autorizzazioni per le politiche di accesso EKS, la gestione del bilanciatore del carico e la pulizia automatica delle risorse del cluster. | 16 novembre 2024 |
| Introdotto the section called "AWS policy gestita: AmazonEKSComputePolicy". | Politica AWS gestita aggiornata di EKS `AmazonEKSComputePolicy`. Autorizzazioni delle risorse aggiornate per l'azione `iam:AddRoleToInstanceProfile`. | 7 novembre 2024 |
| Introdotto the section called "AWS policy gestita: AmazonEKSComputePolicy". | AWS ha introdotto il `AmazonEKSComputePolicy`. | 1 novembre 2024 |
| Aggiunte autorizzazioni a `AmazonEKSClusterPolicy` | Aggiunta l'autorizzazione `ec2:DescribeInstanceTopology` per consentire ad Amazon EKS di allegare informazioni di topologia al nodo come etichette. | 1 novembre 2024 |
| Introdotto AmazonEKS BlockStoragePolicy. | AWS ha introdotto il `AmazonEKSBlockStoragePolicy`. | 30 ottobre 2024 |
| Introdotto the section called "AWS policy gestita: AmazonEKSLoadBalancingPolicy". | AWS ha introdotto il `AmazonEKSLoadBalancingPolicy`. | 30 ottobre 2024 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a AmazonEKSServiceRolePolicy. | Sono state aggiunte `cloudwatch:PutMetricData` autorizzazioni per consentire ad Amazon EKS di pubblicare metriche su Amazon. CloudWatch | 29 ottobre 2024 |
| Introdotto the section called "AWS policy gestita: AmazonEKSNetworkingPolicy". | AWS ha introdotto il. `AmazonEKSNetworkingPolicy` | 28 ottobre 2024 |
| Aggiunte autorizzazioni a `AmazonEKSServicePolicy` e `AmazonEKSServiceRolePolicy` | Aggiunte autorizzazione `ec2:GetSecurityGroupsForVpc` e ai tag associati per consentire a EKS di leggere le informazioni sui gruppi di sicurezza e aggiornare i tag correlati. | 10 ottobre 2024 |
| Introdotto AmazonEKSWorkerNodeMinimalPolicy. | AWS ha introdotto il`AmazonEKSWorkerNodeMinimalPolicy` . | 3 ottobre 2024 |
| Aggiunte autorizzazioni a AWSServiceRoleForAmazonEKSNodegroup. | Sono state aggiunte `autoscaling:ResumeProcesses` le `autoscaling:SuspendProcesses` autorizzazioni per consentire ad Amazon EKS di sospendere e riprendere `AZRebalance` nei gruppi Amazon EKS-managed Auto Scaling. | 21 agosto 2024 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a AWSServiceRoleForAmazonEKSNodegroup. | Aggiunta l'autorizzazione `ec2:DescribeCapacityReservations` per consentire ad Amazon EKS di descrivere la prenotazione della capacità nell'account dell'utente. È stata aggiunta l'autorizzazione `autoscaling:PutScheduledUpdateGroupAction` per abilitare l'impostazione del ridimensionamento pianificato sui gruppi di nodi `CAPACITY_BLOCK` . | 27 giugno 2024 |
| AmazonEKS_CNI_Policy: aggiornamento a una policy esistente | Amazon EKS ha aggiunto nuove autorizzazioni `ec2:DescribeSubnets` per consentire al plugin CNI di Amazon VPC per Kubernetes di visualizzare la quantità di indirizzi IP gratuiti nelle sottoreti Amazon VPC. Il CNI di VPC può utilizzare gli indirizzi IP gratuiti in ciascuna sottorete per scegliere le sottoreti con gli indirizzi IP più liberi da utilizzare durante la creazione di un'interfaccia di rete elastica. | 4 marzo 2024 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| AmazonEKSWorkerNodePolicy: aggiornamento di una policy esistente | Amazon EKS ha aggiunto nuove autorizzazioni per consentire le associazioni EKS Pod Identity. Amazon EKS Pod Identity Agent utilizza il ruolo del nodo. | 26 novembre 2023 |
| Introdotto AmazonEFSCSIDriverPolicy. | AWS ha introdotto il. `AmazonEFSCSIDriverPolicy` | 26 luglio 2023 |
| Aggiunte autorizzazioni a AmazonEKSClusterPolicy. | È stata aggiunta l'autorizzazione `ec2:DescribeAvailabilityZones` per consentire ad Amazon EKS di ottenere i dettagli AZ durante l'individuazione automatica delle sottoreti nella creazione di sistemi di bilanciamento del carico. | 7 febbraio 2023 |
| Condizioni politiche aggiornate in AmazonEBSCSIDriverPolicy. | Sono state rimosse condizioni della policy non valide con caratteri jolly nel campo chiave `StringLike` . È stata aggiunta, inoltre, una nuova condizione `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` a `ec2:DeleteVolume` che consente al driver CSI EBS di eliminare volumi creati dal plugin nella struttura. | 17 novembre 2022 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a AmazonEKSLocalOutpostServiceRolePolicy. | Aggiunti `ec2:DescribeVPCAttribute` , `ec2:GetConsoleOutput` e `ec2:DescribeSecret` per consentire una migliore convalida dei prerequisiti e il controllo gestito del ciclo di vita. Sono stati inoltre aggiunti `ec2:DescribePlacementGroups` , `"arn:aws:ec2:*:*:placement-group/*"` e `ec2:RunInstances` per supportare il controllo del posizionamento delle istanze Amazon EC2 del piano di controllo su Outposts. | 24 ottobre 2022 |
| Aggiorna le autorizzazioni di Amazon Elastic Container Registry in AmazonEKSLocalOutpostClusterPolicy. | L'azione `ecr:GetDownloadUrlForLayer` è stata spostata da tutte le sezioni delle risorse a una sezione con ambito. Aggiunta la risorsa `arn:aws:ecr:*:*:repository/eks/` . Risorsa `arn:aws:ecr:` rimossa. Questa risorsa è coperta dalla risorsa `arn:aws: ecr:*:*:repository/eks/*` aggiunta. | 20 ottobre 2022 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a AmazonEKSLocalOutpostClusterPolicy. | È stato aggiunto il repository `arn:aws:ecr:*:*:repository/kubelet-config-updater` Amazon Elastic Container Registry in modo che le istanze del piano di controllo del cluster possano aggiornare alcuni argomenti `kubelet`. | 31 agosto 2022 |
| Introdotto AmazonEKSLocalOutpostClusterPolicy. | AWS ha introdotto il. `AmazonEKSLocalOutpostClusterPolicy` | 24 agosto 2022 |
| Introdotto AmazonEKSLocalOutpostServiceRolePolicy. | AWS ha introdotto il`AmazonEKSLocalOutpostServiceRolePolicy` . | 23 agosto 2022 |
| Introdotto AmazonEBSCSIDriverPolicy. | AWS ha introdotto il`AmazonEBSCSIDriverPolicy` . | 4 aprile 2022 |
| Aggiunte autorizzazioni a AmazonEKSWorkerNodePolicy. | Aggiunto `ec2:DescribeInstanceTypes` per abilitare le EKS-optimized AMI Amazon in grado di rilevare automaticamente le proprietà a livello di istanza. | 21 marzo 2022 |

| Modifica | Descrizione | Data |
| --- | --- | --- |
| Aggiunte autorizzazioni a AWSServiceRoleForAmazonEKSNodegroup. | Aggiunta l'autorizzazione `autoscaling:EnableMetricsCollection` per consentire ad Amazon EKS l'abilitazione della raccolta di parametri. | 13 dicembre 2021 |
| Aggiunte autorizzazioni a AmazonEKSClusterPolicy. | Aggiunta delle autorizzazioni `ec2:DescribeAccountAttributes`, `ec2:DescribeAddresses`, e `ec2:DescribeInternetGateways` per consentire ad Amazon EKS di creare un ruolo collegato al servizio per un Network Load Balancer (load balancer di Rete). | 17 giugno 2021 |
| Amazon EKS ha iniziato a monitorare le modifiche | Amazon EKS ha iniziato a tracciare le modifiche per le sue politiche AWS gestite. | 17 giugno 2021 |

## Risoluzione dei problemi di IAM

Questo argomento illustra alcuni errori comuni che si potrebbero verificare durante l'utilizzo di Amazon EKS con IAM, e il modo in cui gestirli.

### AccessDeniedException

Se ricevi un messaggio `AccessDeniedException` quando chiami un'operazione AWS API, le credenziali principali IAM che stai utilizzando non dispongono delle autorizzazioni necessarie per effettuare quella chiamata.

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
```

```
User: arn:aws: iam::111122223333:user/user_name is not authorized to perform:
eks:DescribeCluster on resource: arn:aws: eks:region:111122223333:cluster/my-cluster
```

Nel messaggio di esempio precedente, l'utente non dispone delle autorizzazioni per chiamare l'operazione API `DescribeCluster` di Amazon EKS. Per fornire le autorizzazioni da amministratore Amazon EKS a un principale IAM, consultare the section called "Identity-based politiche".

Per informazioni generali su IAM, consultare Controllo dell'accesso tramite policy nella Guida per l'utente di IMA.

## Non riesci a vedere i nodi nella scheda Elaborazione o altro nella scheda Risorse e ricevi un errore nella Console di gestione AWS

È possibile che venga visualizzato un messaggio di errore della console che recita `Your current user or role does not have access to Kubernetes objects on this EKS cluster`. Assicurati che l'utente principale IAM Console di gestione AWS con cui stai utilizzando disponga delle autorizzazioni necessarie. Per ulteriori informazioni, consulta the section called "Autorizzazioni richieste".

## La **`ConfigMap`** per aws-auth non concede l'accesso al cluster

L'Autenticatore AWS IAM non consente un percorso nell'ARN del ruolo utilizzato in `ConfigMap`. Pertanto, rimuovi il percorso prima di specificare `rolearn`. Ad esempio, modifica `arn:aws: iam::111122223333:role/team/developers/eks-admin` in `arn:aws: iam::111122223333:role/eks-admin` .

## Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non disponi delle autorizzazioni a eseguire l'operazione `iam:PassRole`, è necessario aggiornare le policy per poter passare un ruolo ad Amazon EKS.

Alcuni AWS servizi consentono di trasferire un ruolo esistente a quel servizio anziché creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Amazon EKS. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: {arn-aws}iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

### Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon EKS

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per consentire alle persone di accedere alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon EKS supporta queste caratteristiche, consultare the section called "Amazon EKS e IAM".
- Per scoprire come fornire l'accesso alle tue risorse su più AWS account di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro AWS account di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse ad AWS account di terze parti, consulta Fornire l'accesso agli AWS account di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

### I container dei pod riceveranno il seguente errore: **`An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`**

I tuoi contenitori ricevono questo errore se l'applicazione effettua esplicitamente richieste all'endpoint globale AWS STS (`https://sts.amazonaws.com`) e il tuo account di servizio Kubernetes è

configurato per utilizzare un endpoint regionale. Puoi risolvere il problema con una delle opzioni seguenti:

- Aggiorna il codice dell'applicazione per rimuovere le chiamate esplicite all'endpoint globale STS. AWS
- Aggiorna il codice dell'applicazione per effettuare chiamate esplicite agli endpoint regionali, ad esempio `https://sts.us-west-2.amazonaws.com`. L'applicazione dovrebbe avere la ridondanza integrata per scegliere una AWS regione diversa in caso di guasto del servizio nella regione. AWS Per ulteriori informazioni, consulta Managing AWS STS in an AWS Region nella IAM User Guide.
- Configura gli account del servizio per l'utilizzo dell'endpoint globale. I cluster utilizzano l'endpoint regionale per impostazione predefinita. Per ulteriori informazioni, consulta the section called "Endpoint STS".

## Ruolo IAM del cluster Amazon EKS

# richiesto un ruolo IAM del cluster Amazon EKS per ogni cluster. I cluster Kubernetes gestiti da Amazon EKS utilizzano questo ruolo per gestire i nodi e il provider di servizi cloud utilizza questo ruolo per creare bilanciatori del carico con Elastic Load Balancing per i servizi.

Prima di poter creare cluster Amazon EKS, devi creare un ruolo IAM con una delle seguenti policy IAM:

- AmazonEKSClusterPolicy
- Una policy IAM personalizzata. Le autorizzazioni minime che seguono consentono al cluster Kubernetes di gestire i nodi, ma non consentono al Cloud Provider legacy di creare sistemi di bilanciamento del carico con Elastic Load Balancing. La policy IAM personalizzata deve disporre almeno delle seguenti autorizzazioni:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
```

```
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAavailabilityZones",
        "ec2:DescribeInstanceTopology",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Prima del 3 ottobre 2023, AmazonEKSClusterPolicyera richiesto il ruolo IAM per ogni cluster. Prima del 16 aprile 2020, AmazonEKSServicePolicyAmazonEKSClusterPolicyerano obbligatori e il nome suggerito per il ruolo era`eksServiceRole`. Con il ruolo `AWSServiceRoleForAmazonEKS` collegato ai servizi, la AmazonEKSServicePolicypolicy non è più necessaria per i cluster creati a partire dal 16 aprile 2020.

## Verifica della presenza di un ruolo del cluster esistente

Per verificare se l'account dispone già di un ruolo del cluster Amazon EKS, puoi utilizzare la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.

3. Cerca l'elenco dei ruoli per `eksClusterRole`. Se un ruolo che include `eksClusterRole` non esiste, consultare the section called "Creazione del ruolo del cluster Amazon EKS" per crearlo. Se un ruolo che include `eksClusterRole` esiste, seleziona il ruolo per visualizzare le policy allegate.
4. Selezionare Autorizzazioni.
5. Assicurati che la policy AmazonEKSClusterPolicygestita sia associata al ruolo. Se la policy è allegata, il ruolo del cluster Amazon EKS è configurato correttamente.
6. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

## Creazione del ruolo del cluster Amazon EKS

È possibile utilizzare Console di gestione AWS o la AWS CLI per creare il ruolo del cluster.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Scegli Ruoli, quindi Crea ruolo.
c. In Tipo di entità affidabile, seleziona  AWS servizio.
d. Dall'elenco a discesa Casi d'uso per altri AWS servizi, scegli EKS.
e. Scegli EKS - Cluster per il caso d'uso, quindi scegli Successivo.
f. Nella scheda Aggiungi autorizzazioni, scegli Successivo.
g. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `eksClusterRole`.

h. Per Descrizione, inserisci un testo descrittivo come `Amazon EKS - Cluster role`.

i. Scegli Crea ruolo.

AWS CLI

a. Copiare i seguenti contenuti in un file denominato *`cluster-trust-policy.json`*.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crea il ruolo. Puoi sostituire *`eksClusterRole`* con un nome a tua scelta.

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

c. Allega la policy IAM richiesta al ruolo.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name eksClusterRole
```

## Ruolo IAM del nodo Amazon EKS

Il `kubelet` daemon del nodo Amazon EKS effettua chiamate alle AWS API per tuo conto. I nodi ricevono le autorizzazioni per queste chiamate API attraverso un profilo dell'istanza IAM e le policy associate. Prima di avviare i nodi e registrarli in un cluster, devi creare un ruolo IAM che i nodi possano utilizzare all'avvio. Questo requisito si applica ai nodi di lavoro avviati con l'AMI ottimizzata per Amazon EKS fornita da Amazon o con altre AMI di nodi che intendi utilizzare. Inoltre, questo requisito si applica sia ai gruppi di nodi gestiti sia ai nodi autogestiti.

> **Note**
>
> Non è possibile utilizzare lo stesso ruolo utilizzato per creare i cluster.

Prima di creare i nodi, è necessario creare un ruolo IAM con le seguenti autorizzazioni:

- Autorizzazioni per consentire a `kubelet` di descrivere le risorse Amazon EC2 nel VPC, come quelle fornite dalla policy AmazonEKSWorkerNodePolicy. Questa policy fornisce anche le autorizzazioni per il Pod Identity Agent di Amazon EKS.
- Autorizzazioni per consentire a `kubelet` di utilizzare le immagini dei container da Amazon Elastic Container Registry (Amazon ECR), come previsto dalla policy AmazonEC2ContainerRegistryPullOnly. Le autorizzazioni per utilizzare le immagini dei container da Amazon Elastic Container Registry (Amazon ECR) sono necessarie perché i componenti aggiuntivi integrati per le reti eseguono pod che utilizzano immagini di container provenienti da Amazon ECR.
- (Facoltativo) Autorizzazioni per consentire al Pod Identity Agent di Amazon EKS di utilizzare l'azione `eks-auth:AssumeRoleForPodIdentity` per recuperare le credenziali per i pod. Se non utilizzi il AmazonEKSWorkerNodePolicy, devi fornire questa autorizzazione in aggiunta alle autorizzazioni EC2 per utilizzare EKS Pod Identity.
- (Facoltativo) Se non si usa Pod Identity di EKS o IRSA per fornire le autorizzazioni ai pod VPC CNI, è necessario fornire le autorizzazioni per il VPC CNI sul ruolo dell'istanza. È possibile utilizzare la policy gestita `AmazonEKS_CNI_Policy` (se il cluster è stato creato con la famiglia `IPv4`) o una policy IPv6 creata (se il cluster è stato creato con la famiglia `IPv6`). Invece di allegare la policy a questo ruolo, tuttavia, consigliamo di allegarla a un ruolo separato utilizzato specificamente per il componente aggiuntivo CNI di Amazon VPC. Per ulteriori informazioni sulla creazione di un ruolo separato per il componente aggiuntivo CNI di Amazon VPC, consulta the section called "Configurare per IRSA".
- (Facoltativo) Autorizzazioni `kubelet` per l'utilizzo delle immagini dei contenitori di Amazon ECR Public (public.ecr.aws), come fornite dalla policy. AmazonElasticContainerRegistryPublicReadOnly Queste autorizzazioni sono consigliate se si estraggono immagini di contenitori da ECR Public. Senza di esse, le estrazioni avranno successo, ma potrebbero avere una frequenza limitata.

> **Note**
>
> I gruppi di nodi Amazon EC2 devono avere un ruolo IAM diverso dal profilo Fargate. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".

## Verifica della presenza di un ruolo di nodo esistente

Per controllare se l'account dispone già di un ruolo di nodo Amazon EKS, utilizza la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Ricerca `eksNodeRole`, `AmazonEKSNodeRole` o `NodeInstanceRole` nell'elenco di ruoli. Se un ruolo con uno di questi nomi non esiste, consulta the section called "Creazione del ruolo IAM del nodo Amazon EKS" per creare il ruolo. Se esiste un ruolo che contiene `eksNodeRole`, `AmazonEKSNodeRole` o `NodeInstanceRole`, seleziona il ruolo per visualizzare le policy allegate.
4. Seleziona Autorizzazioni.
5. Assicurati che AmazonEKSWorkerNodePolicy le policy AmazonEC2ContainerRegistryPullOnly gestite siano collegate al ruolo o che sia allegata una policy personalizzata con le autorizzazioni minime.

   > **Note**
   >
   > Se la policy AmazonEKS_CNI_Policy è allegata al ruolo, è consigliabile invece rimuoverla e allegarla a un ruolo IAM mappato all'account di servizio `aws-node` Kubernetes. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

6. Scegli Trust relationships (Relazioni di attendibilità), quindi scegli Edit trust policy (Modifica policy di attendibilità).
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

## Creazione del ruolo IAM del nodo Amazon EKS

Puoi creare il ruolo IAM del nodo con Console di gestione AWS o la AWS CLI.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Nel pannello di navigazione a sinistra, seleziona Ruoli.
c. Nella pagina Ruoli, seleziona Crea ruolo.
d. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
    i. Nella sezione Tipo di entità attendibile, scegli AWS il servizio.
    ii. Per Caso d'uso, seleziona EC2.
    iii. Scegli Successivo.
e. Nella pagina Aggiungi autorizzazioni, collega una policy personalizzata o esegui le operazioni seguenti:
    i. Nella casella Filtra policy, inserisci `AmazonEKSWorkerNodePolicy`.
    ii. Seleziona la casella di controllo a sinistra della AmazonEKSWorkerNodePolicy nei risultati di ricerca.
    iii. Scegli Cancella filtri.
    iv. Nella casella Filtra policy, inserisci `AmazonEC2ContainerRegistryPullOnly`.
    v. Seleziona la casella di controllo a sinistra della AmazonEC2ContainerRegistryPullOnly nei risultati di ricerca.

Inoltre, a questo ruolo o a un ruolo diverso mappato all'account del servizio Kubernetes `aws-node` deve essere collegata la policy gestita AmazonEKS_CNI_Policy o una policy IPv6 creata da te. Si consiglia di assegnare la policy al ruolo associato all'account del servizio Kubernetes anziché assegnarlo a questo ruolo. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

vi. Scegli Avanti.

f. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:

i. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSNodeRole`.

ii. In Descrizione, sostituisci il testo attuale con un testo descrittivo come `Amazon EKS - Node role`.

iii. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.

iv. Scegli Crea ruolo.

AWS CLI

a. Per creare il file `node-role-trust-relationship.json`, emetti il seguente comando:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

b. Crea il ruolo IAM.

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

c. Allegare al ruolo IAM le due policy gestite IAM richieste.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly \
  --role-name AmazonEKSNodeRole
```

d. Allega una delle seguenti policy IAM al ruolo IAM a seconda della famiglia di IP con cui hai creato il cluster. La policy deve essere allegata a questo ruolo o a un ruolo associato all'account di servizio Kubernetes `aws-node` che viene utilizzato per il componente aggiuntivo Amazon VPC CNI per Kubernetes. Si consiglia di assegnare la policy al ruolo associato all'account del servizio Kubernetes. Per assegnare la policy al ruolo associato all'account del servizio Kubernetes, consultare the section called "Configurare per IRSA".

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSNodeRole
```

- IPv6

A. Copia il testo seguente e salvalo in un file denominato `vpc-cni-ipv6-policy.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

B. Creare la policy IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

C. Allega la policy IAM al ruolo IAM. Sostituisci *111122223333* con l'ID del tuo account.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

## Ruolo IAM del cluster della modalità automatica di Amazon EKS

# richiesto un ruolo IAM del cluster di Amazon EKS per ogni cluster. I cluster di Kubernetes gestiti da Amazon EKS utilizzano questo ruolo per automatizzare le attività di routine per lo storage, il networking e il dimensionamento automatico di elaborazione.

Prima di poter creare cluster Amazon EKS, devi creare un ruolo IAM con le policy richieste per la modalità automatica di EKS. Puoi allegare le AWS policy gestite IAM suggerite o creare policy personalizzate con autorizzazioni equivalenti.

- AmazonEKSComputePolicy
- AmazonEKSBlockStoragePolicyV2
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy

- AmazonEKSClusterPolicy

## Personalizzato AWS tag per le risorse EKS Auto

Per impostazione predefinita, le politiche gestite relative a EKS Auto Mode non consentono l'applicazione di tag definiti dall'utente alle AWS risorse assegnate alla modalità automatica. Se si desidera applicare tag definiti dall'utente alle AWS risorse, è necessario allegare autorizzazioni aggiuntive al ruolo IAM del Cluster con autorizzazioni sufficienti per creare e modificare i tag sulle risorse. AWS Di seguito è riportato un esempio di policy che consentirà l'accesso ai tag senza restrizioni:

Visualizzare un esempio di policy sui tag personalizzata

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Compute",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateFleet",
                "ec2:RunInstances",
                "ec2:CreateLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-node-class-name": "*",
                    "aws:RequestTag/eks:kubernetes-node-pool-name": "*"
                }
            }
        },
        {
            "Sid": "Storage",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVolume",
                "ec2:CreateSnapshot"
```

```
            ],
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:snapshot/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "Networking",
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-cni-node-name": "*"
                }
            }
        },
        {
            "Sid": "LoadBalancer",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateRule",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
```

```
        },
        {
            "Sid": "ShieldProtection",
            "Effect": "Allow",
            "Action": [
                "shield:CreateProtection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldTagResource",
            "Effect": "Allow",
            "Action": [
                "shield:TagResource"
            ],
            "Resource": "arn:aws:shield::*:protection/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        }
    ]
}
```

## Verifica della presenza di un ruolo del cluster esistente

Per verificare se l'account dispone già di un ruolo del cluster Amazon EKS, puoi utilizzare la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca l'elenco dei ruoli per `AmazonEKSAutoClusterRole`. Se un ruolo che include `AmazonEKSAutoClusterRole` non esiste, consultare le istruzioni nella sezione successiva per

crearlo. Se un ruolo che include `AmazonEKSAutoClusterRole` esiste, seleziona il ruolo per visualizzare le policy allegate.

4. Selezionare Autorizzazioni.
5. Assicurati che la policy  AmazonEKSClusterPolicy  gestita sia associata al ruolo. Se la policy è allegata, il ruolo del cluster Amazon EKS è configurato correttamente.
6. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

> **Note**
>
> AWS non richiede il nome `AmazonEKSAutoClusterRole` per questo ruolo.

## Creazione del ruolo del cluster Amazon EKS

È possibile utilizzare Console di gestione AWS o la AWS CLI per creare il ruolo del cluster.

Console di gestione AWS

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.

2. Scegli Ruoli, quindi Crea ruolo.
3. In Tipo di entità attendibile, seleziona il AWS servizio.
4. Dall'elenco a discesa Casi d'uso per altri AWS servizi, scegli EKS.
5. Scegli EKS - Cluster per il caso d'uso, quindi scegli Successivo.
6. Nella scheda Aggiungi autorizzazioni, selezionare le policy e quindi scegliere Avanti.
   - AmazonEKSComputePolicy
   - AmazonEKSBlockStoragePolicyV2
   - AmazonEKSLoadBalancingPolicy
   - AmazonEKSNetworkingPolicy
   - AmazonEKSClusterPolicy
7. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSAutoClusterRole`.
8. Per Descrizione, inserisci un testo descrittivo come `Amazon EKS - Cluster role`.
9. Scegli Crea ruolo.

AWS CLI

1. Copiare i seguenti contenuti in un file denominato *`cluster-trust-policy.json`*.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. Crea il ruolo. Puoi sostituire *`AmazonEKSAutoClusterRole`* con un nome a tua scelta.

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. Allegare le policy IAM richieste al ruolo:

AmazonEKSClusterPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicyV2:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicyV2
```

AmazonEKSLoadBalancingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

# Ruolo IAM del nodo della modalità automatica di Amazon EKS

> **Note**
>
> Non puoi utilizzare lo stesso ruolo usato per creare i cluster.

Prima di creare i nodi, devi creare un ruolo IAM con le seguenti policy, o autorizzazioni equivalenti:

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

## Verifica della presenza di un ruolo di nodo esistente

Per controllare se l'account dispone già di un ruolo di nodo Amazon EKS, utilizza la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca l'elenco dei ruoli per `AmazonEKSAutoNodeRole`. Se un ruolo con uno di questi nomi non esiste, consulta le istruzioni nella sezione successiva per creare il ruolo. Se esiste un ruolo che contiene `AmazonEKSAutoNodeRole`, selezionare il ruolo per visualizzare i criteri allegati.
4. Seleziona Autorizzazioni.
5. Assicurati che siano allegate le policy richieste sopra riportate o le policy personalizzate equivalenti.
6. Scegli Relazioni di attendibilità, quindi scegli Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

## Creazione del ruolo IAM del nodo Amazon EKS

Puoi creare il ruolo IAM del nodo con Console di gestione AWS o la AWS CLI.

Console di gestione AWS

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Nella sezione Tipo di entità affidabile, scegli  AWS servizio.
   b. Per Caso d'uso, seleziona EC2.
   c. Scegli Successivo.
5. Nella pagina Aggiungi autorizzazioni, allega le policy seguenti:
   - AmazonEKSWorkerNodeMinimalPolicy
   - AmazonEC2ContainerRegistryPullOnly
6. Nella pagina Assegna un nome, rivedi e crea, esegui le operazioni seguenti:
   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSAutoNodeRole`.
   b. In Descrizione, sostituisci il testo attuale con un testo descrittivo come `Amazon EKS - Node role`.
   c. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   d. Scegli Crea ruolo.

AWS CLI

Crea il ruolo IAM del nodo

Utilizza il node-trust-policyfile.json del passaggio precedente per definire quali entità possono assumere il ruolo. Per creare il ruolo IAM del nodo, esegui il seguente comando:

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Annota l'ARN del ruolo

Una volta creato il ruolo, recupera e salva l'ARN del ruolo IAM del nodo. Sarà necessario usare questo ARN nei passaggi successivi. Usa il seguente comando per ottenere l'ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Allega le policy richieste

Allega le seguenti policy AWS gestite al ruolo Node IAM per fornire le autorizzazioni necessarie:

Per collegare Amazon EKSWorkerNodeMinimalPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Per collegare Amazon EC2ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Allega politica opzionale

Se desideri estrarre immagini di container da Public ECR, devi allegare la seguente policy AWS gestita al ruolo Node IAM per garantire che i nodi possano autenticarsi su Public ECR ed estrarre le immagini senza limitazioni.

Da allegare: AmazonElasticContainerRegistryPublicReadOnly

```
aws iam attach-role-policy \
```

```
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonElasticContainerRegistryPublicReadOnly
```

# Funzionalità Amazon EKS, ruolo IAM

Le funzionalità EKS richiedono la configurazione di un ruolo IAM di capacità (o ruolo di capacità). Le funzionalità utilizzano questo ruolo per eseguire azioni sui AWS servizi e accedere alle risorse Kubernetes nel cluster tramite voci di accesso create automaticamente.

Prima di poter specificare un ruolo di capacità durante la creazione della capacità, è necessario creare il ruolo IAM con la policy di fiducia e le autorizzazioni appropriate per il tipo di funzionalità. Una volta creato, questo ruolo IAM può essere riutilizzato per un numero qualsiasi di risorse funzionali.

## Requisiti del ruolo di capacità

Il ruolo di capacità deve soddisfare i seguenti requisiti:

- Il ruolo deve appartenere allo stesso AWS account del cluster e della risorsa di capacità
- Il ruolo deve avere una politica di fiducia che consenta al servizio EKS Capabilities di assumerlo
- Il ruolo deve disporre delle autorizzazioni appropriate per il tipo di capacità e i requisiti del caso d'uso (vedithe section called “Autorizzazioni per tipo di funzionalità”)

## Politica di fiducia per i ruoli di capacità

Tutti i ruoli di capacità devono includere la seguente politica di fiducia:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
```

```
}
```

Questa politica di fiducia consente a EKS di:

- Assumi il ruolo di eseguire operazioni AWS API
- Tagga le sessioni per scopi di controllo e tracciamento

## Autorizzazioni per tipo di funzionalità

Le autorizzazioni IAM richieste dipendono dalla funzionalità utilizzata e dal modello di implementazione.

> **Note**
>
> Per le implementazioni di produzione che utilizzano IAM Role Selectors con ACK o quando si utilizza kro o Argo CD senza integrazione di AWS servizi, Capability Role potrebbe non richiedere alcuna autorizzazione IAM oltre alla policy di fiducia.

kro (Kube Resource Orchestrator)

Non sono richieste autorizzazioni IAM. Puoi creare un ruolo di capacità senza policy allegate. kro richiede solo le autorizzazioni RBAC di Kubernetes per creare e gestire le risorse Kubernetes.

AWS Controller per Kubernetes (ACK)

ACK supporta due modelli di autorizzazione:

- Configurazione semplice (sviluppo/test): aggiungi le autorizzazioni di AWS servizio direttamente al Capability Role. È la soluzione ideale per iniziare, per le implementazioni con account singolo o quando tutti gli utenti necessitano delle stesse autorizzazioni.
- Best practice di produzione: utilizza IAM Role Selectors per implementare l'accesso con privilegi minimi. Con questo approccio, il Capability Role necessita solo dell'`sts:AssumeRole`autorizzazione per assumere ruoli specifici del servizio. Non si aggiungono autorizzazioni di AWS servizio (come S3 o RDS) al ruolo di capacità stesso: tali autorizzazioni vengono concesse a singoli ruoli IAM mappati su namespace specifici.

  I selettori di ruolo IAM consentono di:

  - Isolamento delle autorizzazioni a livello di namespace

- Gestione delle risorse tra account
- Ruoli IAM specifici del team
- Modello di sicurezza con privilegi minimi

  Esempio di policy Capability Role per l'approccio IAM Role Selector:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::111122223333:role/ACK-S3-Role",
        "arn:aws:iam::111122223333:role/ACK-RDS-Role",
        "arn:aws:iam::444455556666:role/ACKCrossAccountRole"
      ]
    }
  ]
}
```

  Per una configurazione dettagliata delle autorizzazioni ACK, inclusi IAM Role Selectors, consulta. the section called “Configurazione delle autorizzazioni ”

CD Argo

Per impostazione predefinita, non sono richieste autorizzazioni IAM. Potrebbero essere necessarie autorizzazioni opzionali per:

- AWS Secrets Manager: se si utilizza Secrets Manager per archiviare le credenziali del repository Git
- AWS CodeConnections: Se si utilizza CodeConnections per l'autenticazione del repository Git

  Politica di esempio per Secrets Manager e CodeConnections:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
```

```
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:region:account-id:secret:argocd/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection",
        "codeconnections:GetConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/*"
    }
  ]
}
```

Per i requisiti di autorizzazione dettagliati di Argo CD, vederethe section called "Considerazioni su Argo CD".

## Verificate la presenza di un ruolo di capacità esistente

Puoi utilizzare la seguente procedura per verificare se il tuo account dispone già di un ruolo IAM con funzionalità adatto al tuo caso d'uso.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca nell'elenco dei ruoli il nome del tuo ruolo di capacità (ad esempio, `ACKCapabilityRole` o`ArgoCDCapabilityRole`).
4. Se esiste un ruolo, selezionalo per visualizzare le politiche allegate e la relazione di fiducia.
5. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
6. Verifica che la relazione di fiducia corrisponda alla politica di fiducia in materia di capacità. Se non corrisponde, aggiorna la politica di fiducia.
7. Scegli Autorizzazioni e verifica che il ruolo disponga delle autorizzazioni appropriate per il tipo di capacità e il caso d'uso.

## Creazione di una funzionalità (ruolo IAM)

È possibile utilizzare Console di gestione AWS o la AWS CLI per creare un ruolo di capacità.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Scegli Ruoli, quindi Crea ruolo.
c. In Tipo di entità affidabile, seleziona Politica di fiducia personalizzata.
d. Copia e incolla la politica di attendibilità delle capacità nell'editor dei criteri di fiducia.
e. Scegli Next (Successivo).
f. Nella scheda Aggiungi autorizzazioni, seleziona o crea politiche appropriate per il tuo tipo di capacità (vedithe section called "Autorizzazioni per tipo di funzionalità"). Per kro, puoi saltare questo passaggio.
g. Scegli Next (Successivo).
h. Per Nome ruolo, inserisci un nome univoco per il tuo ruolo, ad esempio `ACKCapabilityRoleArgoCDCapabilityRole`, o. `kroCapabilityRole`
i. Per Descrizione, inserisci un testo descrittivo come `Amazon EKS - ACK capability role`.
j. Scegliere Crea ruolo.

AWS CLI

a. Copia la policy di affidabilità delle capacità in un file denominato`capability-trust-policy.json`.
b. Crea il ruolo. Sostituiscila `ACKCapabilityRole` con il nome del ruolo desiderato.

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://capability-trust-policy.json
```

c. Allega le politiche IAM richieste al ruolo. Per ACK, allega le policy per AWS i servizi che desideri gestire. Per Argo CD, allegate le politiche per Secrets Manager o CodeConnections se necessario. Per kro, puoi saltare questo passaggio.

   Esempio per ACK con autorizzazioni S3:

```
aws iam put-role-policy \
  --role-name ACKCapabilityRole \
  --policy-name S3Management \
  --policy-document file://s3-policy.json
```

## Risoluzione dei problemi relativi alle funzionalità e ai ruoli

La creazione delle funzionalità fallisce con «Ruolo IAM non valido»

Verificare che:

- Il ruolo esiste nello stesso account del cluster
- La politica di fiducia corrisponde alla politica di affidabilità delle capacità
- Hai l'`iam:PassRole`autorizzazione per il ruolo

La funzionalità mostra errori di autorizzazione

Verificare che:

- Il ruolo dispone delle autorizzazioni IAM necessarie per il tipo di funzionalità
- La voce di accesso per il ruolo esiste nel cluster
- Se necessario, vengono configurate autorizzazioni Kubernetes aggiuntive (vedi) the section called "Autorizzazioni Kubernetes aggiuntive"

Le risorse ACK falliscono con errori di «autorizzazione negata»

Verificare che:

- Il ruolo dispone delle autorizzazioni IAM necessarie per il tuo caso d'uso
- Per i controller ACK che fanno riferimento a segreti, assicurati di aver associato la policy di `AmazonEKSSecretReaderPolicy` accesso prevista ai namespace appropriati.

Per ulteriori indicazioni sulla risoluzione dei problemi, consulta. the section called "Considerazioni sulle funzionalità EKS"

# Monitoraggio delle prestazioni del cluster e visualizzazione dei log

Amazon EKS mette a disposizione una serie di strumenti di monitoraggio o registrazione per l'osservazione dei dati. I dati di log di Amazon EKS possono essere trasmessi in streaming a AWS servizi o a strumenti partner per l'analisi dei dati. Sono disponibili molti servizi Console di gestione AWS che forniscono dati per la risoluzione dei problemi di Amazon EKS. Puoi anche utilizzare una soluzione open source AWS supportata per monitorare l'infrastruttura Amazon EKS.

Dopo aver selezionato Cluster nel riquadro di navigazione sinistro della console Amazon EKS, puoi visualizzare l'integrità e i dettagli del cluster selezionandone il nome e scegliendo la scheda Osservabilità. Per visualizzare i dettagli delle risorse Kubernetes esistenti implementate nel cluster, consulta the section called "Accesso alle risorse di cluster".

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon EKS e delle tue AWS soluzioni. Ti consigliamo di raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione. per consentire un debug più facile di eventuali errori in più punti. Prima di iniziare il monitoraggio di Amazon EKS, è opportuno creare un piano di monitoraggio che tenga conto dei seguenti aspetti.

- Quali sono gli obiettivi? Sono necessarie notifiche in tempo reale se i cluster si ridimensionano notevolmente?
- Di quali risorse si intende eseguire il monitoraggio?
- Con quale frequenza sarà eseguito il monitoraggio di queste risorse? L'azienda vuole rispondere rapidamente ai rischi?
- Quali strumenti verranno utilizzati? Se utilizzi già AWS Fargate come parte del lancio, puoi utilizzare il log router integrato.
- Chi eseguirà le attività di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

## Registrazione e monitoraggio su Amazon EKS

Amazon EKS offre strumenti integrati per la registrazione e il monitoraggio. Per le versioni supportate, la dashboard di osservabilità offre visibilità sulle prestazioni del cluster. Ti aiuta a rilevare, risolvere e

correggere rapidamente i problemi. Oltre alle funzionalità di monitoraggio, include elenchi basati sui log di audit del piano di controllo (control-plane). Il piano di controllo di Kubernetes espone una serie di metriche che possono essere analizzate anche all'esterno della console.

I log del piano di controllo (control-plane) registrano tutte le chiamate dell'API ai tuoi cluster, le informazioni di controllo che acquisiscono quali utenti hanno eseguito determinate azioni sui tuoi cluster, e informazioni basate sui ruoli. Per ulteriori informazioni, consulta Registrazione e monitoraggio su Amazon EKS nella Guida AWS prescrittiva.

La registrazione del piano di controllo di Amazon EKS fornisce log di audit e diagnostica direttamente dal piano di controllo di Amazon CloudWatch EKS ai log del tuo account. Questi log consentono di semplificare la protezione e l'esecuzione dei cluster. Puoi selezionare i tipi di log esatti di cui hai bisogno e i log vengono inviati come flussi di log a un gruppo per ogni cluster Amazon EKS in cui opera. CloudWatch Per ulteriori informazioni, consulta the section called “Registrazione dei log del piano di controllo (control-plane)”.

> **Note**
>
> Quando controlli i log dell'autenticatore di Amazon EKS su Amazon CloudWatch, vengono visualizzate le voci che contengono testo simile al seguente testo di esempio.
>
> ```
> level=info msg="mapping IAM role" groups="[]" role="arn:aws:
>  iam::111122223333:role/XXXXXXXXXXXXXXXXX-NodeManagerRole-XXXXXXXX"
>  username="eks:node-manager"
> ```
>
> Sono previste voci che contengono questo testo. Lo `username` è un ruolo di servizio Amazon EKS interno che esegue operazioni specifiche per i gruppi di nodi gestiti e nodi Fargate. Per la registrazione personalizzabile di basso livello, consulta Registrazione Kubernetes.

Amazon EKS è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in Amazon EKS. CloudTrail acquisisce tutte le chiamate API per Amazon EKS come eventi. Le chiamate acquisite includono le chiamate dalla console Amazon EKS e le chiamate di codice alle operazioni API Amazon EKS. Per ulteriori informazioni, consulta the section called “ AWS CloudTrail”.

Il server API Kubernetes espone una serie di parametri che sono utili per il monitoraggio e l'analisi dei dati. Per ulteriori informazioni, consulta the section called “Parametri Prometheus”.

Per configurare Fluent Bit per CloudWatch i log Amazon personalizzati, consulta Configurazione di Fluent Bit nella Amazon CloudWatch  User Guide.

# Strumenti di registrazione e monitoraggio di Amazon EKS

Amazon Web Services offre vari strumenti che permettono di monitorare Amazon EKS. Alcuni di questi strumenti possono essere configurati in modo che eseguano automaticamente il monitoraggio, mentre altri richiedono l'intervento manuale. Ti consigliamo di automatizzare il più possibile i processi di monitoraggio nella misura in cui l'ambiente e il set di strumenti esistenti lo consentono.

La tabella seguente descrive varie opzioni dello strumento di monitoraggio.

| Aree | Strumento | Description | Configurazione |
|---|---|---|---|
| Piano di controllo | Dashboard di osservabilità | Per le versioni supportate, la dashboard di osservabilità offre visibilità sulle prestazioni del cluster. Ti aiuta a rilevare, risolvere e corregger e rapidamente i problemi. | Procedura di configurazione |
| Applicazioni/piano di controllo (control-plane) | Prometheus | Prometheus può essere utilizzato per monitorare metriche e avvisi per le applicazi oni e il piano di controllo (control-plane). | Procedura di configurazione |
| Applicazioni | CloudWatch Container Insights | CloudWatch Container Insights raccoglie, aggrega e riepiloga metriche e log delle applicazi | Procedura di configurazione |

| Aree | Strumento | Description | Configurazione |
| --- | --- | --- | --- |
| | | oni e dei microservizi containerizzati. | |
| Applicazioni | AWS OpenTelemetry Distro per (ADOT) | ADOT può raccoglie re e inviare metriche, dati di traccia e metadati correlati a servizi o partner di monitoraggio. AWS Può essere configura to tramite Container Insights. CloudWatch | Procedura di configurazione |
| Applicazioni | Amazon DevOps Guru | Amazon DevOps Guru rileva le prestazioni operative e la disponibilità a livello di nodo. | Procedura di configurazione |
| Applicazioni | AWS X-Ray | AWS X-Ray riceve dati di traccia sulla tua applicazione. Questi dati di traccia includono le richieste in entrata e in uscita e i metadati relativi a tali richieste. Per Amazon EKS, l'impleme ntazione richiede il OpenTelemetry componente aggiuntiv o. | Procedura di configurazione |

| Aree | Strumento | Description | Configurazione |
| --- | --- | --- | --- |
| Applicazioni | Amazon CloudWatch | CloudWatch fornisce gratuitamente alcuni parametri di base di Amazon EKS nelle versioni supportat e. Puoi espandere questa funzional ità con CloudWatc h Observability Operator per gestire la raccolta di metriche, log e dati di tracciamento. | Procedura di configurazione |

La tabella seguente descrive varie opzioni dello strumento di registrazione.

| Aree | Strumento | Description | Configurazione |
| --- | --- | --- | --- |
| Piano di controllo | Dashboard di osservabilità | Per le versioni supportate, la dashboard di osservabilità mostra elenchi basati sui log di audit del piano di controllo (control-plane). Include anche collegamenti per controllare i log dei piani in Amazon CloudWatch. | Procedura di configurazione |
| Applicazioni | Amazon CloudWatch Container Insights | Amazon CloudWatc h Container Insights raccoglie, aggrega e | Procedura di configurazione |

| Aree | Strumento | Description | Configurazione |
| --- | --- | --- | --- |
| | | riepiloga i parametri e i log delle tue applicazioni e microservizi containerizzati. | |
| Piano di controllo (control-plane) | CloudWatch Registri Amazon | Puoi inviare i log di controllo e diagnostica direttamente dal piano di controllo di Amazon CloudWatch EKS ai registri del tuo account. | Procedura di configurazione |
| Piano di controllo (control-plane) | AWS CloudTrail | Registra le chiamate API da parte di un utente, un ruolo o un servizio. | Procedura di configurazione |
| Diverse aree per le istanze AWS Fargate | AWS Router di registro Fargate | Per le istanze AWS Fargate, il router di log trasmette i log ai servizi o agli AWS strumenti dei partner. Utilizza AWS per Fluent Bit. I log possono essere trasmessi in streaming ad altri servizi o strumenti partner. AWS | Procedura di configurazione |

# Monitora il tuo cluster con la dashboard di osservabilità

La console Amazon EKS include una dashboard di osservabilità che offre visibilità nelle prestazioni del cluster. Le informazioni che fornisce aiutano a rilevare, risolvere e chiarire rapidamente i problemi. È possibile aprire la sezione applicabile del dashboard di osservabilità scegliendo un elemento nel Riepilogo di integrità e prestazioni. Questo riepilogo è incluso in diversi punti, inclusa la scheda Osservabilità.

La dashboard di osservabilità è suddivisa in diverse schede.

## Riepilogo

Il riepilogo dello stato e delle prestazioni elenca la quantità di articoli in varie categorie. Ciascun numero funge da collegamento ipertestuale a una posizione nella dashboard di osservabilità con un elenco per quella categoria.

## Integrità del cluster

L'integrità del cluster fornisce notifiche importanti di cui essere a conoscenza, su alcune delle quali potrebbe essere necessario intervenire il prima possibile. Con questo elenco, è possibile vedere le descrizioni e le risorse interessate. Lo stato del cluster include due tabelle: Problemi di integrità e Approfondimenti sulla configurazione. Per aggiornare lo stato dei Problemi di integrità, scegli il pulsante di aggiornamento ( ↻ ). Gli Approfondimenti sulla configurazione si aggiornano automaticamente una volta ogni 24 ore e non possono essere aggiornate manualmente.

Per ulteriori informazioni sui Problemi di integrità, consulta the section called "Domande frequenti sull'integrità del cluster e codici di errore con percorsi di risoluzione". Per ulteriori informazioni su Approfondimenti sulla configurazione, consulta the section called "Approfondimenti sui cluster".

## Monitoraggio del piano di controllo

La scheda Monitoraggio del piano di controllo è divisa in tre sezioni, ognuna delle quali aiuta a monitorare e risolvere i problemi del piano di controllo del cluster.

### Metriche

Per i cluster con versione Kubernetes `1.28` e successive, la sezione Metriche mostra grafici di diverse metriche raccolte per vari componenti del piano di controllo.

Puoi impostare il periodo X-axis di tempo utilizzato da ogni grafico effettuando le selezioni nella parte superiore della sezione. È possibile aggiornare i dati con il pulsante di aggiornamento ( ↻ ). Per ogni grafico separato, il pulsante con le ellissi verticali (⋮) apre un menu con le opzioni di. CloudWatch

Queste e altre metriche sono automaticamente disponibili come metriche di monitoraggio di base nel CloudWatch namespace. `AWS/EKS` Per ulteriori informazioni, consulta la sezione Monitoraggio di base e monitoraggio dettagliato nella Amazon CloudWatch User Guide. Per ottenere metriche, visualizzazioni e approfondimenti più dettagliati, consulta Container Insights nella Amazon CloudWatch User Guide. Oppure, se si preferisce un monitoraggio basato su Prometheus, consulta the section called "Parametri Prometheus".

La tabella seguente descrive le metriche disponibili.

| Metrica | Description |
| --- | --- |
| Richieste APIServer | Le richieste al minuto effettuate al server API. |
| Richieste totali 4XX del server API | Il numero di richieste al minuto del server API con codici di risposta HTTP 4XX (errori lato client). |
| Richieste totali 5XX del server API | Il numero di richieste al minuto del server API con codici di risposta HTTP 5XX (errori lato server). |
| Richieste totali 429 del server API | Il numero di richieste al minuto del server API con codici di risposta HTTP 429 (troppe richieste). |
| Dimensioni dell'archiviazione | La dimensione del database di archiviazione (`etcd`). |
| Tentativi del sistema di pianificazione | Il numero di tentativi di pianificare i pod in base ai risultati «unschedulable», «error» e «scheduled». |
| Pod in sospeso | Il numero di pod in sospeso per tipo di coda di "attivo", "backoff", "non pianificabile" e "limitato". |

| Metrica | Description |
| --- | --- |
| Latenza delle richieste del server API | La latenza per le richieste del server API. |
| Richieste correnti del server API | Le richieste correnti per il server API. |
| Richieste webhook | Le richieste webhook al minuto. |
| Rifiuti delle richieste webhook | Il numero di richieste webhook che sono state rifiutate. |
| Latenza della richiesta webhook P99 | La latenza del 99° percentile delle richieste webhook esterne di terze parti. |

## CloudWatch Log Insights

La sezione CloudWatch Log Insights mostra vari elenchi basati sui log di controllo del piano di controllo. I log del piano di controllo di Amazon EKS devono essere attivati per utilizzare questa funzionalità, che puoi eseguire dalla sezione Visualizza i log del piano di controllo. CloudWatch

Quando è trascorso abbastanza tempo per raccogliere i dati, è possibile Eseguire tutte le query o scegliere Esegui query per un singolo elenco alla volta. Verranno addebitati costi aggiuntivi a partire da CloudWatch ogni richiesta. Nella parte superiore della sezione, scegli il periodo di tempo dei risultati che desideri visualizzare. Se desideri un controllo più avanzato per qualsiasi query, puoi scegliere Visualizza in. CloudWatch Ciò ti consentirà di aggiornare una query in CloudWatch base alle tue esigenze.

Per ulteriori informazioni, consulta Analyzing log data with CloudWatch Logs Insights nella Amazon CloudWatch Logs User Guide.

## Visualizza i log del piano di controllo in CloudWatch

Per aggiornare i tipi di log disponibili, scegli Gestisci log. Dopo aver abilitato la registrazione, occorrono alcuni minuti prima che i log vengano visualizzati in CloudWatch Logs. Quando è trascorso tempo sufficiente, scegli uno dei link Visualizza in questa sezione per accedere al log applicabile.

Per ulteriori informazioni, consulta the section called “Registrazione dei log del piano di controllo (control-plane)”.

## Approfondimenti sui cluster

La tabella Migliora approfondimenti fa emergere i problemi e consiglia azioni correttive, accelerando il processo di convalida per l'aggiornamento a nuove versioni di Kubernetes. Amazon EKS esegue automaticamente la scansione dei cluster e li mette a confronto con un elenco di potenziali problemi di aggiornamento della versione Kubernetes. La tabella Migliora approfondimenti elenca i controlli approfonditi eseguiti da Amazon EKS su questo cluster, insieme ai relativi stati associati.

Amazon EKS gestisce e aggiorna periodicamente l'elenco dei controlli degli approfondimenti da eseguire sulla base delle valutazioni delle modifiche nel progetto Kubernetes e delle modifiche al servizio Amazon EKS legate alle nuove versioni. La console Amazon EKS aggiorna automaticamente lo stato di ciascun approfondimento, che può essere visualizzato nella colonna relativa all'ora dell'ultimo aggiornamento.

Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

## Problemi di integrità dei nodi

L'agente di monitoraggio dei nodi Amazon EKS legge automaticamente i log dei nodi per rilevare problemi di integrità. Indipendentemente dall'impostazione di riparazione automatica, vengono segnalati tutti i problemi di integrità dei nodi in modo da poterli esaminare se necessario. Se un tipo di problema è elencato senza una descrizione, è possibile leggere la descrizione nel relativo elemento popover.

Quando si aggiorna la pagina, tutti i problemi risolti scompariranno dall'elenco. Se la riparazione automatica è abilitata, potresti visualizzare temporaneamente alcuni problemi di integrità che verranno risolti senza intervento da parte dell'utente. I problemi non supportati dalla riparazione automatica potrebbero richiedere un'azione manuale da parte dell'utente a seconda del tipo.

Per segnalare problemi di integrità dei nodi, il cluster deve utilizzare la modalità automatica Amazon EKS o disporre del componente aggiuntivo dell'agente di monitoraggio dei nodi. Per ulteriori informazioni, consulta the section called "Salute e riparazione dei nodi".

## Funzionalità EKS

La sezione Capacità mostra lo stato e lo stato delle risorse EKS Capability nel cluster. Le notifiche sullo stato e sullo stato di entrambe le funzionalità e le relative risorse Kubernetes gestite nel cluster possono essere monitorate qui. Quando si aggiorna la pagina, tutti i problemi risolti scompariranno dall'elenco.

Per ulteriori informazioni, consulta the section called "Lavorare con le funzionalità".

# Monitora il traffico dei carichi di lavoro Kubernetes con Container Network Observability

Amazon EKS offre funzionalità avanzate di osservabilità della rete che forniscono informazioni più approfondite sull'ambiente di rete dei container. Queste funzionalità ti aiutano a comprendere, monitorare e risolvere meglio il tuo panorama di rete Kubernetes in. AWS Con una migliore osservabilità della rete di container, puoi sfruttare metriche granulari relative alla rete per un migliore rilevamento proattivo delle anomalie nel traffico del cluster, nei flussi cross-AZ e nei servizi. AWS Utilizzando queste metriche, puoi misurare le prestazioni del sistema e visualizzare le metriche sottostanti utilizzando il tuo stack di osservabilità preferito.

Inoltre, Amazon EKS ora fornisce visualizzazioni di monitoraggio della rete nella AWS console che accelerano e migliorano la risoluzione precisa dei problemi per un'analisi più rapida delle cause principali. Puoi anche sfruttare queste funzionalità visive per individuare i principali interlocutori e i flussi di rete che causano ritrasmissioni e timeout di ritrasmissione, eliminando i punti ciechi durante gli incidenti.

Queste funzionalità sono abilitate da Amazon CloudWatch Network Flow Monitor.

## Casi d'uso

### Misura le prestazioni della rete per rilevare anomalie

Diversi team si basano su uno stack di osservabilità che consente loro di misurare le prestazioni del sistema, visualizzare le metriche del sistema e allarmarsi nel caso in cui venga superata una soglia specifica. L'osservabilità della rete di container in EKS si allinea a questo aspetto, esponendo le metriche chiave del sistema che è possibile analizzare per ampliare l'osservabilità delle prestazioni di rete del sistema a livello di pod e nodo di lavoro.

### Sfrutta le visualizzazioni della console per una risoluzione dei problemi più precisa

In caso di allarme proveniente dal sistema di monitoraggio, potresti voler concentrarti sul cluster e sul carico di lavoro da cui ha avuto origine il problema. A tale scopo, puoi sfruttare le visualizzazioni della console EKS che restringono l'ambito di indagine a livello di cluster e accelerano la divulgazione dei flussi di rete responsabili della maggior parte delle ritrasmissioni, dei timeout di ritrasmissione e del volume di dati trasferiti.

### Tieni traccia dei migliori oratori nel tuo ambiente Amazon EKS

Molti team utilizzano EKS come base per le proprie piattaforme, rendendolo il punto focale dell'attività di rete di un ambiente applicativo. Utilizzando le funzionalità di monitoraggio della rete di questa funzionalità, puoi monitorare quali carichi di lavoro sono responsabili della maggior parte del traffico (misurato in base al volume di dati) all'interno del cluster, tra le AZ, nonché del traffico verso destinazioni esterne all'interno ( AWS DynamoDB e S3) e oltre il AWS cloud (Internet o on-premise). Inoltre, è possibile monitorare le prestazioni di ciascuno di questi flussi in base alle ritrasmissioni, ai timeout di ritrasmissione e ai dati trasferiti.

## Funzionalità

1. Metriche delle prestazioni: questa funzionalità consente di acquisire le metriche di sistema relative alla rete per pod e nodi di lavoro direttamente dall'agente Network Flow Monitor (NFM) in esecuzione nel cluster EKS.
2. Mappa dei servizi: questa funzionalità visualizza dinamicamente l'intercomunicazione tra i carichi di lavoro nel cluster, consentendo di divulgare rapidamente le metriche chiave (ritrasmissioni - RT, timeout di ritrasmissione - RTO e dati trasferiti - DT) associate ai flussi di rete tra pod comunicanti.
3. Tabella dei flussi: con questa tabella, puoi monitorare i migliori oratori nei carichi di lavoro Kubernetes del tuo cluster da tre diverse angolazioni: visualizzazione del AWS servizio, visualizzazione del cluster e visualizzazione esterna. Per ogni vista, puoi vedere le ritrasmissioni, i timeout di ritrasmissione e i dati trasferiti tra il pod di origine e la sua destinazione.
   - AWS visualizzazione del servizio: mostra i migliori oratori ai AWS servizi (DynamoDB e S3)
   - Visualizzazione cluster: mostra i migliori oratori all'interno del cluster (da est ← a → ovest)
   - Vista esterna: mostra i migliori oratori alle destinazioni esterne al cluster all'esterno AWS

## Nozioni di base

Per iniziare, abilita Container Network Observability nella console EKS per un cluster nuovo o esistente. Ciò automatizzerà la creazione delle dipendenze di Network Flow Monitor (NFM) (risorse Scope and Monitor). Inoltre, dovrai installare il componente aggiuntivo Network Flow Monitor Agent. In alternativa, puoi installare queste dipendenze utilizzando le API EKS (per il componente aggiuntivo) `AWS CLI`, le APINFM o Infrastructure as Code (come Terraform). Una volta stabilite queste dipendenze, puoi configurare il tuo strumento di monitoraggio preferito per raccogliere le metriche delle prestazioni di rete per pod e nodi di lavoro dall'agente NFM. Per visualizzare l'attività di

rete e le prestazioni dei carichi di lavoro, è possibile accedere alla console EKS nella scheda «Rete» del dashboard di osservabilità del cluster.

Quando si utilizza Network Flow Monitor in EKS, è possibile mantenere il flusso di lavoro e lo stack tecnologico di osservabilità esistenti, sfruttando al contempo una serie di funzionalità aggiuntive che consentono di comprendere e ottimizzare ulteriormente il livello di rete del proprio ambiente EKS. Puoi saperne di più sui prezzi di Network Flow Monitor qui.

## Prerequisiti e note importanti

1. Come accennato in precedenza, se abilitate Container Network Observability dalla console EKS, le dipendenze delle risorse NFM sottostanti (Scope e Monitor) verranno create automaticamente per vostro conto e sarete guidati attraverso il processo di installazione del componente aggiuntivo EKS per NFM.
2. Se desideri abilitare questa funzionalità utilizzando Infrastructure as Code (IaC) come Terraform, dovrai definire le seguenti dipendenze nel tuo IaC: NFM Scope, NFM Monitor, componente aggiuntivo EKS per NFM. Inoltre, dovrai concedere le autorizzazioni pertinenti al componente aggiuntivo EKS utilizzando Pod Identity o IAM roles for service accounts (IRSA).
3. È necessario eseguire una versione minima di 1.1.0 per il componente aggiuntivo EKS dell'agente NFM.
4. È necessario utilizzare la versione 6.21.0 o successiva di Terraform AWS Provider per il supporto delle risorse di Network Flow Monitor.

### Autorizzazioni IAM richieste

Componente aggiuntivo EKS per l'agente NFM

Puoi utilizzare la policy `CloudWatchNetworkFlowMonitorAgentPublishPolicy`AWS gestita con Pod Identity. Questa policy contiene le autorizzazioni per l'agente NFM a inviare report di telemetria (metriche) a un endpoint Network Flow Monitor.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "networkflowmonitor:Publish"
      ],
```

```
      "Resource" : "*"
    }
  ]
}
```

Osservabilità della rete dei container nella console EKS

Le seguenti autorizzazioni sono necessarie per abilitare la funzionalità e visualizzare la mappa dei servizi e la tabella dei flussi nella console.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect": "Allow",
      "Action": [
        "networkflowmonitor:ListScopes",
        "networkflowmonitor:ListMonitors",
        "networkflowmonitor:GetScope",
        "networkflowmonitor:GetMonitor",
        "networkflowmonitor:CreateScope",
        "networkflowmonitor:CreateMonitor",
        "networkflowmonitor:TagResource",
        "networkflowmonitor:StartQueryMonitorTopContributors",
        "networkflowmonitor:StopQueryMonitorTopContributors",
        "networkflowmonitor:GetQueryStatusMonitorTopContributors",
        "networkflowmonitor:GetQueryResultsMonitorTopContributors"
      ],
      "Resource": "*"
    }
  ]
}
```

## Utilizzo di AWS CLI, API EKS e API NFM

```
#!/bin/bash

# Script to create required Network Flow Monitor resources
set -e

CLUSTER_NAME="my-eks-cluster"
CLUSTER_ARN="arn:aws:eks:{Region}:{Account}:cluster/{ClusterName}"
```

```
REGION="us-west-2"
AGENT_NAMESPACE="amazon-network-flow-monitor"

echo "Creating Network Flow Monitor resources..."

# Check if Network Flow Monitor agent is running in the cluster
echo "Checking for Network Flow Monitor agent in cluster..."
if kubectl get pods -n "$AGENT_NAMESPACE" --no-headers 2>/dev/null | grep -q "Running";
 then
    echo "Network Flow Monitor agent exists and is running in the cluster"
else
    echo "Network Flow Monitor agent not found. Installing as EKS addon..."
    aws eks create-addon \
        --cluster-name "$CLUSTER_NAME" \
        --addon-name "$AGENT_NAMESPACE" \
        --region "$REGION"
    echo "Network Flow Monitor addon installation initiated"
fi

# Get Account ID
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)

echo "Cluster ARN: $CLUSTER_ARN"
echo "Account ID: $ACCOUNT_ID"

# Check for existing scope
echo "Checking for existing Network Flow Monitor Scope..."
EXISTING_SCOPE=$(aws networkflowmonitor list-scopes --region $REGION --query
 'scopes[0].scopeArn' --output text 2>/dev/null || echo "None")

if [ "$EXISTING_SCOPE" != "None" ] && [ "$EXISTING_SCOPE" != "null" ]; then
    echo "Using existing scope: $EXISTING_SCOPE"
    SCOPE_ARN=$EXISTING_SCOPE
else
    echo "Creating new Network Flow Monitor Scope..."
    SCOPE_RESPONSE=$(aws networkflowmonitor create-scope \
        --targets "[{\"targetIdentifier\":{\"targetId\":{\"accountId\":
\"${ACCOUNT_ID}\"},\"targetType\":\"ACCOUNT\"},\"region\":\"${REGION}\"}]" \
        --region $REGION \
        --output json)

    SCOPE_ARN=$(echo $SCOPE_RESPONSE | jq -r '.scopeArn')
    echo "Scope created: $SCOPE_ARN"
fi
```

```
# Create Network Flow Monitor with EKS Cluster as local resource
echo "Creating Network Flow Monitor..."
MONITOR_RESPONSE=$(aws networkflowmonitor create-monitor \
    --monitor-name "${CLUSTER_NAME}-monitor" \
    --local-resources "type=AWS::EKS::Cluster,identifier=${CLUSTER_ARN}" \
    --scope-arn "$SCOPE_ARN" \
    --region $REGION \
    --output json)

MONITOR_ARN=$(echo $MONITOR_RESPONSE | jq -r '.monitorArn')

echo "Monitor created: $MONITOR_ARN"

echo "Network Flow Monitor setup complete!"
echo "Monitor ARN: $MONITOR_ARN"
echo "Scope ARN: $SCOPE_ARN"
echo "Local Resource: AWS::EKS::Cluster (${CLUSTER_ARN})"
```

## Utilizzo dell'infrastruttura come codice (IaC)

### Terraform

Se utilizzi Terraform per gestire la tua infrastruttura AWS cloud, puoi includere le seguenti configurazioni di risorse per abilitare l'osservabilità della rete di contenitori per il tuo cluster.

#### Ambito NFM

```
data "aws_caller_identity" "current" {}

resource "aws_networkflowmonitor_scope" "example" {
  target {
    region = "us-east-1"
    target_identifier {
      target_type = "ACCOUNT"
      target_id {
        account_id = data.aws_caller_identity.current.account_id
      }
    }
  }

  tags = {
    Name = "example"
```

```
  }
}
```

Monitor NFM

```
resource "aws_networkflowmonitor_monitor" "example" {
  monitor_name = "eks-cluster-name-monitor"
  scope_arn    = aws_networkflowmonitor_scope.example.scope_arn

  local_resource {
    type       = "AWS::EKS::Cluster"
    identifier = aws_eks_cluster.example.arn
  }

  remote_resource {
    type       = "AWS::Region"
    identifier = "us-east-1" # this must be the same region that the cluster is in
  }

  tags = {
    Name = "example"
  }
}
```

Componente aggiuntivo EKS per NFM

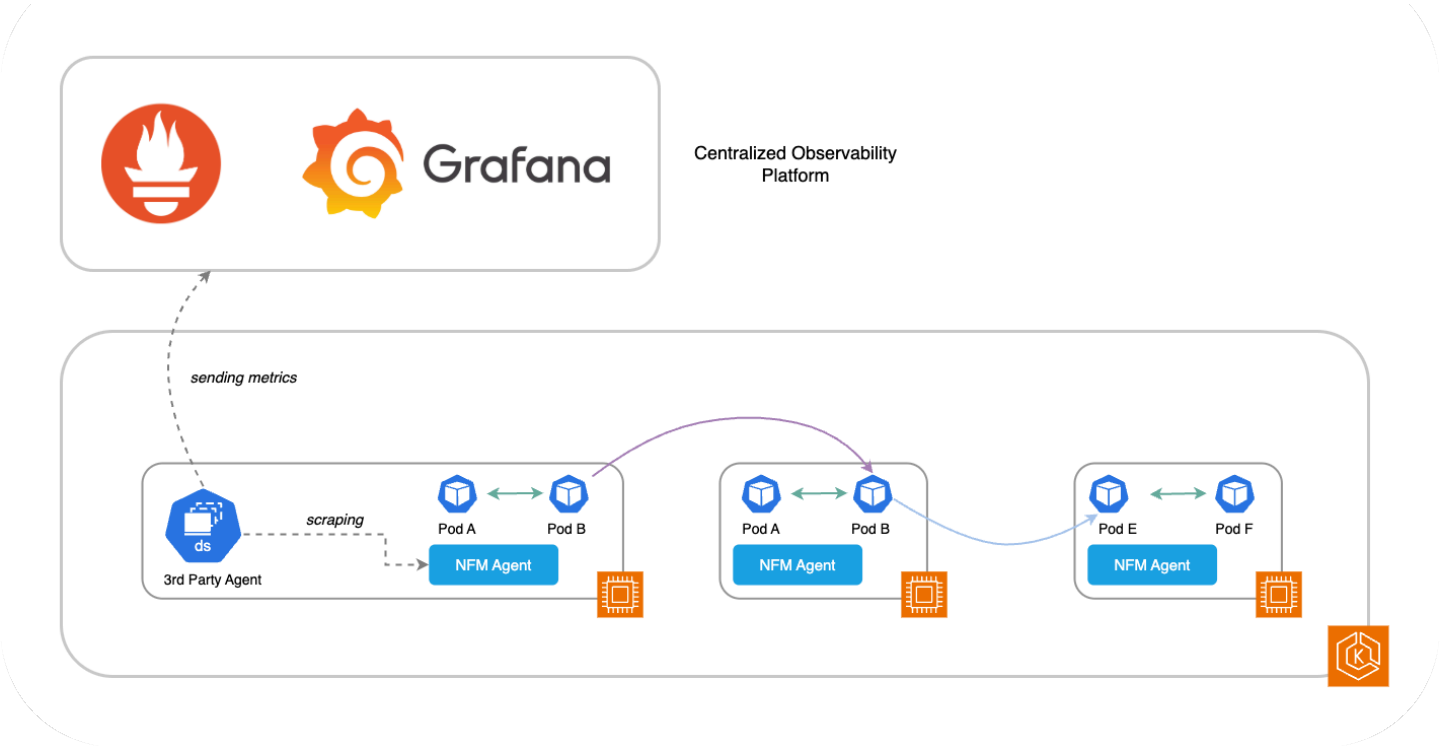
```
resource "aws_eks_addon" "example" {
  cluster_name                = aws_eks_cluster.example.name
  addon_name                  = "aws-network-flow-monitoring-agent"
}
```

# Come funziona?

## Metriche delle prestazioni

### Parametri del sistema

Se utilizzi strumenti di terze parti (3P) per monitorare il tuo ambiente EKS (come Prometheus e Grafana), puoi acquisire le metriche di sistema supportate direttamente dall'agente Network Flow Monitor. Queste metriche possono essere inviate allo stack di monitoraggio per ampliare la misurazione delle prestazioni di rete del sistema a livello di pod e nodo di lavoro. Le metriche disponibili sono elencate nella tabella, in Metriche di sistema supportate.



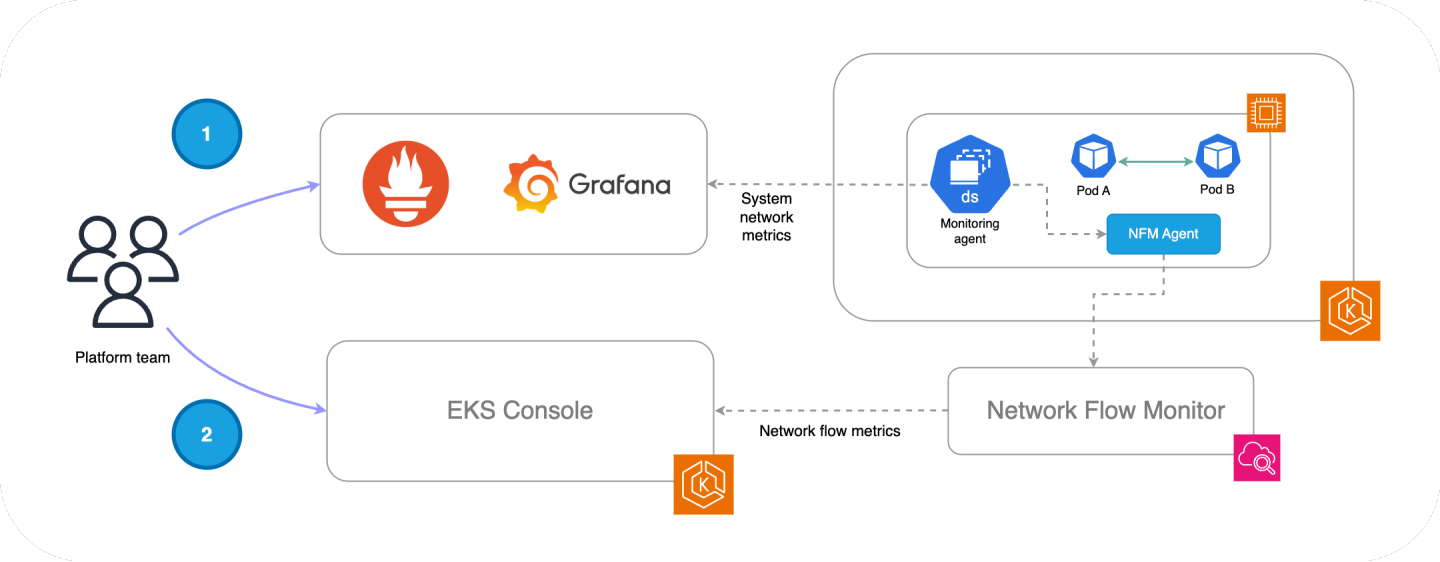

Per abilitare queste metriche, sovrascrivete le seguenti variabili di ambiente utilizzando le variabili di configurazione durante il processo di installazione (vedi:): https://aws.amazon.com/blogs/containers/amazon-eks-add-ons-advanced-configuration/

```
OPEN_METRICS:
    Enable or disable open metrics. Disabled if not supplied
    Type: String
    Values: [“on”, “off”]
OPEN_METRICS_ADDRESS:
    Listening IP address for open metrics endpoint. Defaults to 127.0.0.1 if not
 supplied
    Type: String
OPEN_METRICS_PORT:
    Listening port for open metrics endpoint. Defaults to 80 if not supplied
    Type: Integer
    Range: [0..65535]
```

Metriche a livello di flusso

Inoltre, Network Flow Monitor acquisisce i dati del flusso di rete insieme alle metriche del livello di flusso: ritrasmissioni, timeout di ritrasmissione e dati trasferiti. Questi dati vengono elaborati da Network Flow Monitor e visualizzati nella console EKS per far emergere il traffico nell'ambiente del cluster e il relativo rendimento in base a queste metriche del livello di flusso.

Il diagramma seguente illustra un flusso di lavoro in cui entrambi i tipi di metriche (sistema e livello di flusso) possono essere sfruttati per ottenere una maggiore intelligenza operativa.




1. Il team della piattaforma può raccogliere e visualizzare le metriche di sistema nel proprio stack di monitoraggio. Una volta attivati gli avvisi, possono rilevare anomalie di rete o problemi che influiscono sui pod o sui nodi di lavoro utilizzando le metriche di sistema fornite dall'agente NFM.
2. Come passaggio successivo, i team della piattaforma possono sfruttare le visualizzazioni native della console EKS per restringere ulteriormente l'ambito delle indagini e accelerare la risoluzione dei problemi in base alle rappresentazioni dei flussi e alle metriche associate.

Nota importante: l'acquisizione delle metriche di sistema dall'agente NFM e il processo con cui l'agente NFM invia le metriche a livello di flusso al backend NFM sono processi indipendenti.

Metriche di sistema supportate

Nota importante: le metriche di sistema vengono esportate in formato. OpenMetrics

| Nome parametro | Tipo | Dimensioni | Description |
|---|---|---|---|
| ingress_flow | Gauge | instance_id, iface, pod, namespace, nodo | Conteggio del flusso TCP in ingresso () TcpPassiveOpens |

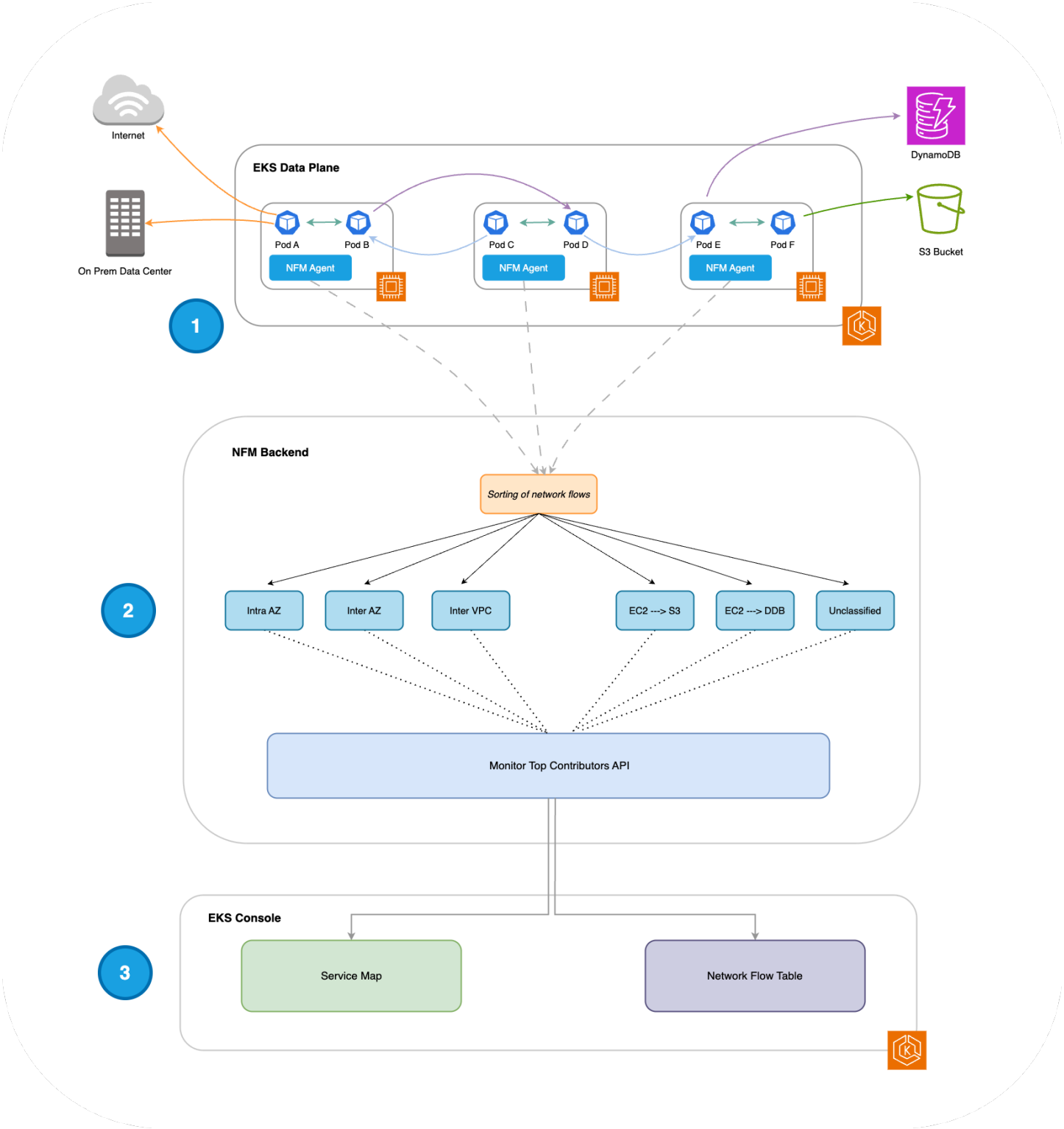
| Nome parametro | Tipo | Dimensioni | Description |
|---|---|---|---|
| Flusso_uscita | Gauge | instance_id, iface, pod, namespace, node | Conteggio del flusso TCP in uscita () TcpActiveOpens |
| pacchetti_in ingresso | Gauge | instance_id, iface, pod, namespace, nodo | Numero di pacchetti in ingresso (delta) |
| pacchetti_in uscita | Gauge | instance_id, iface, pod, namespace, node | Numero di pacchetti in uscita (delta) |
| ingress_bytes | Gauge | instance_id, iface, pod, namespace, nodo | Conteggio dei byte in ingresso (delta) |
| egress_bytes | Gauge | instance_id, iface, pod, namespace, nodo | Conteggio dei byte in uscita (delta) |
| bw_in_allowance_exceeded | Gauge | instance_id, eni, nodo | Pacchetti queued/dropped dovuti al limite di larghezza di banda in entrata |
| bw_out_allowance_exceeded | Gauge | instance_id, eni, nodo | Pacchetti queued/dropped dovuti al limite di larghezza di banda in uscita |
| pps_allowance_exceeded | Gauge | instance_id, eni, nodo | Pacchetti queued/dropped dovuti al limite PPS bidirezionale |

| Nome parametro | Tipo | Dimensioni | Description |
| --- | --- | --- | --- |
| conntrack_allowanc e_exceeded | Gauge | instance_id, eni, nodo | Pacchetti persi a causa del limite di tracciamento della connessione |
| linklocal_allowanc e_exceeded | Gauge | instance_id, eni, nodo | I pacchetti sono caduti a causa del limite PPS del servizio proxy locale |

Metriche del livello di flusso supportate

| Nome parametro | Tipo | Description |
| --- | --- | --- |
| Ritrasmissioni TCP | Contatore | Numero di volte in cui un mittente invia nuovamente un pacchetto perso o danneggia to durante la trasmissione. |
| Timeout di ritrasmissione TCP | Contatore | Numero di volte in cui un mittente ha avviato un periodo di attesa per determinare se un pacchetto è andato perso durante il transito. |
| Dati (byte) trasferiti | Contatore | Volume di dati trasferiti tra un'origine e una destinazione per un determinato flusso. |

# Mappa dei servizi e tabella dei flussi



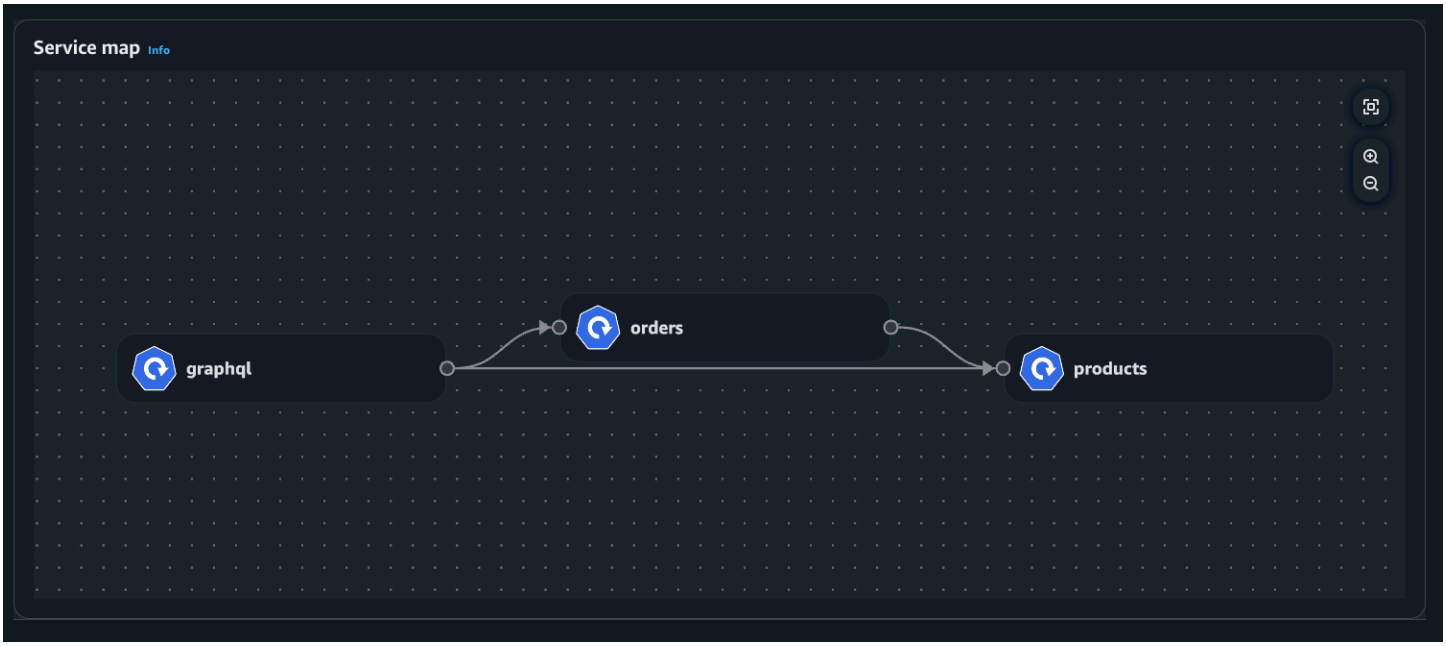

1. Una volta installato, l'agente Network Flow Monitor viene eseguito DaemonSet su ogni nodo di lavoro e raccoglie i primi 500 flussi di rete (in base al volume di dati trasferiti) ogni 30 secondi.

2. Questi flussi di rete sono suddivisi nelle seguenti categorie: Intra AZ, Inter AZ, EC2 → S3, EC2 → DynamoDB (DDB) e Unclassified. A ogni flusso sono associate 3 metriche: ritrasmissioni, timeout di ritrasmissione e dati trasferiti (in byte).
   - Intra AZ: flussi di rete tra pod nella stessa AZ
   - Inter AZ: flussi di rete tra pod in diverse AZ
   - EC2 → S3: la rete scorre dai pod a S3
   - EC2 → DDB: flussi di rete dai pod al DDB
   - Non classificato: la rete scorre dai pod a Internet o in locale
3. I flussi di rete dell'API Network Flow Monitor Top Contributors vengono utilizzati per potenziare le seguenti esperienze nella console EKS:
   - Mappa dei servizi: visualizzazione dei flussi di rete all'interno del cluster (Intra AZ e Inter AZ).
   - Tabella dei flussi: presentazione in tabella dei flussi di rete all'interno del cluster (Intra AZ e Inter AZ), dai pod ai AWS servizi (EC2 → S3 ed EC2 → DDB) e dai pod alle destinazioni esterne (non classificate).

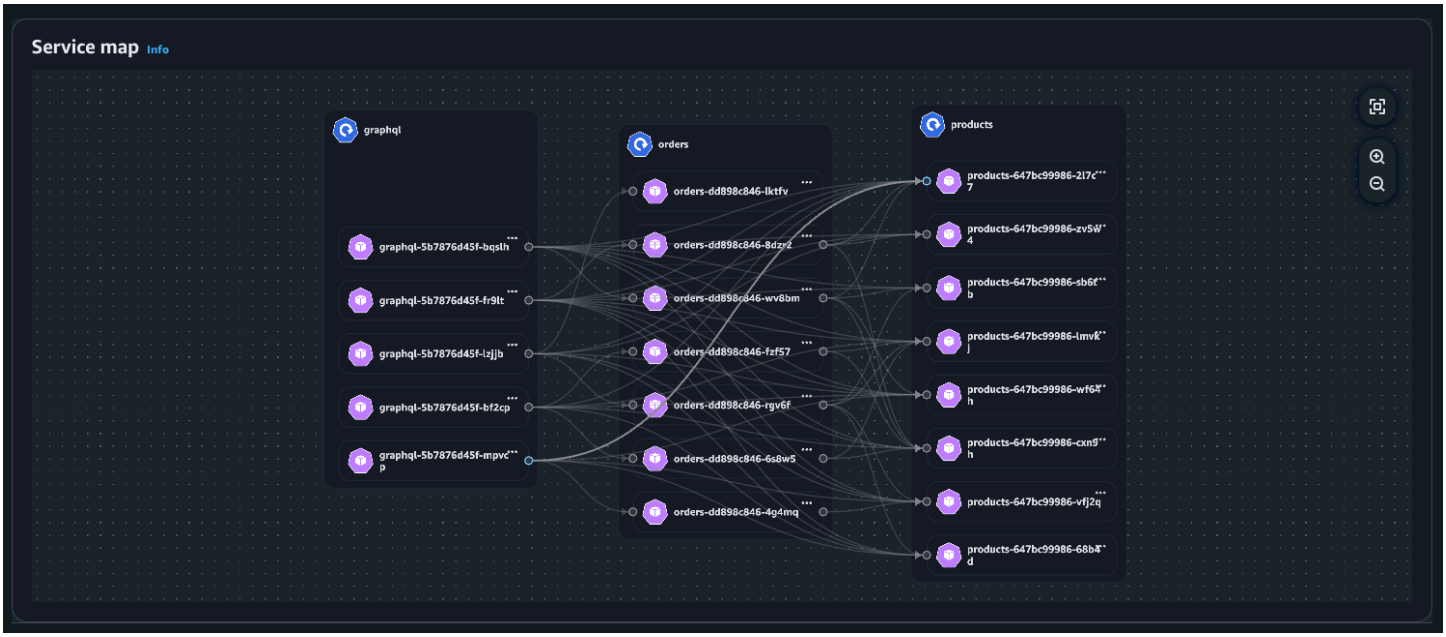
I flussi di rete estratti dall'API Top Contributors sono limitati a un intervallo di tempo di 1 ora e possono includere fino a 500 flussi per ciascuna categoria. Per quanto riguarda la mappa dei servizi, ciò significa che è possibile reperire e presentare fino a 1000 flussi dalle categorie di flusso Intra AZ e Inter AZ in un intervallo di tempo di 1 ora. Per quanto riguarda la tabella dei flussi, ciò significa che è possibile reperire e presentare fino a 2500 flussi di rete da tutte e 5 le categorie di flussi di rete in un intervallo di tempo di 2 ore.

Esempio: mappa dei servizi

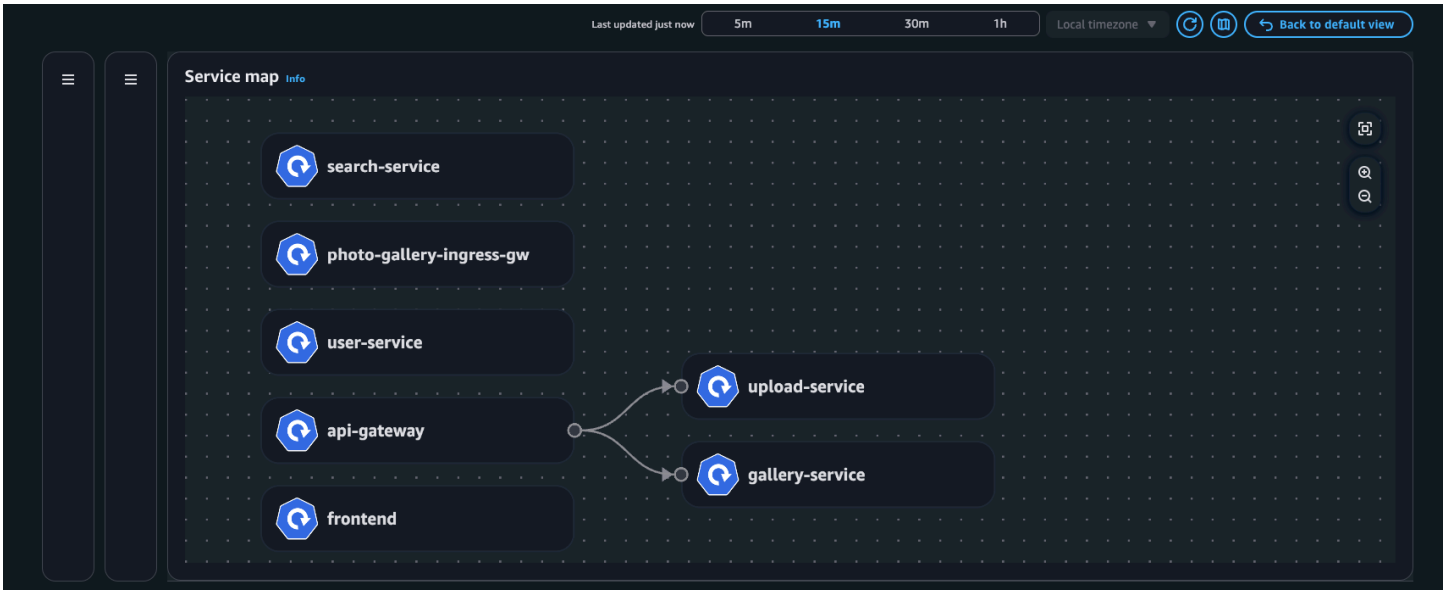
Visualizzazione della distribuzione



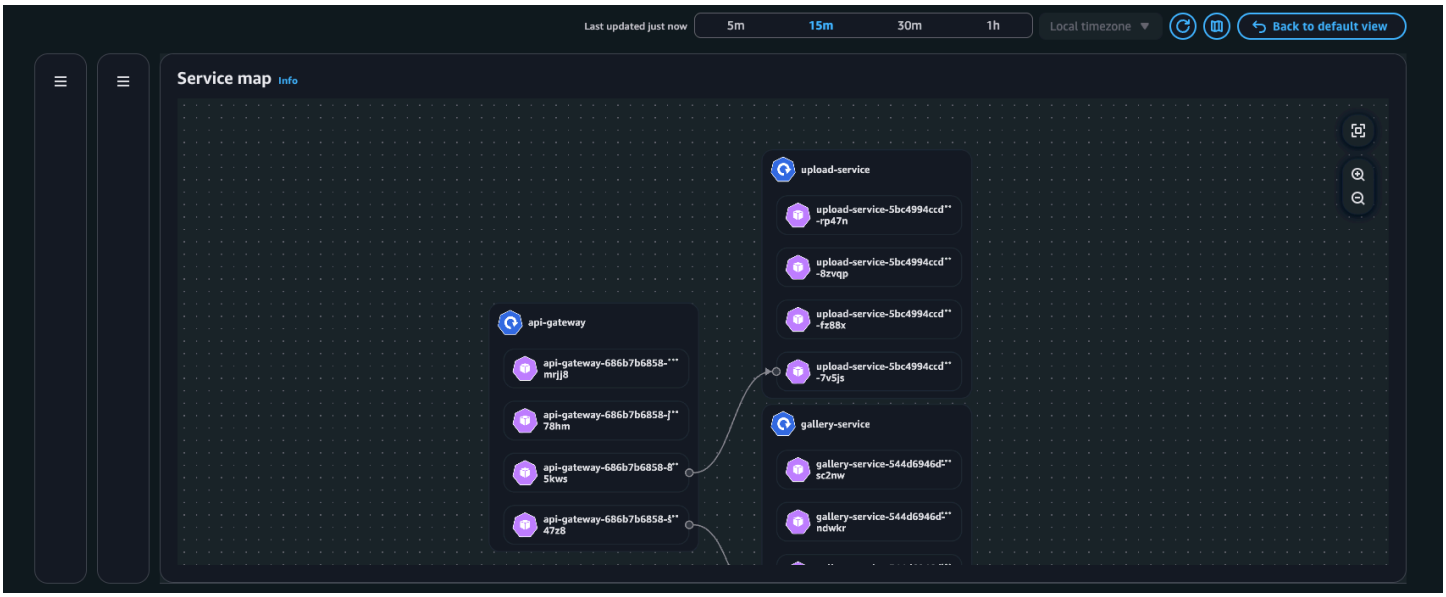

## Vista Pod




## Vista di distribuzione

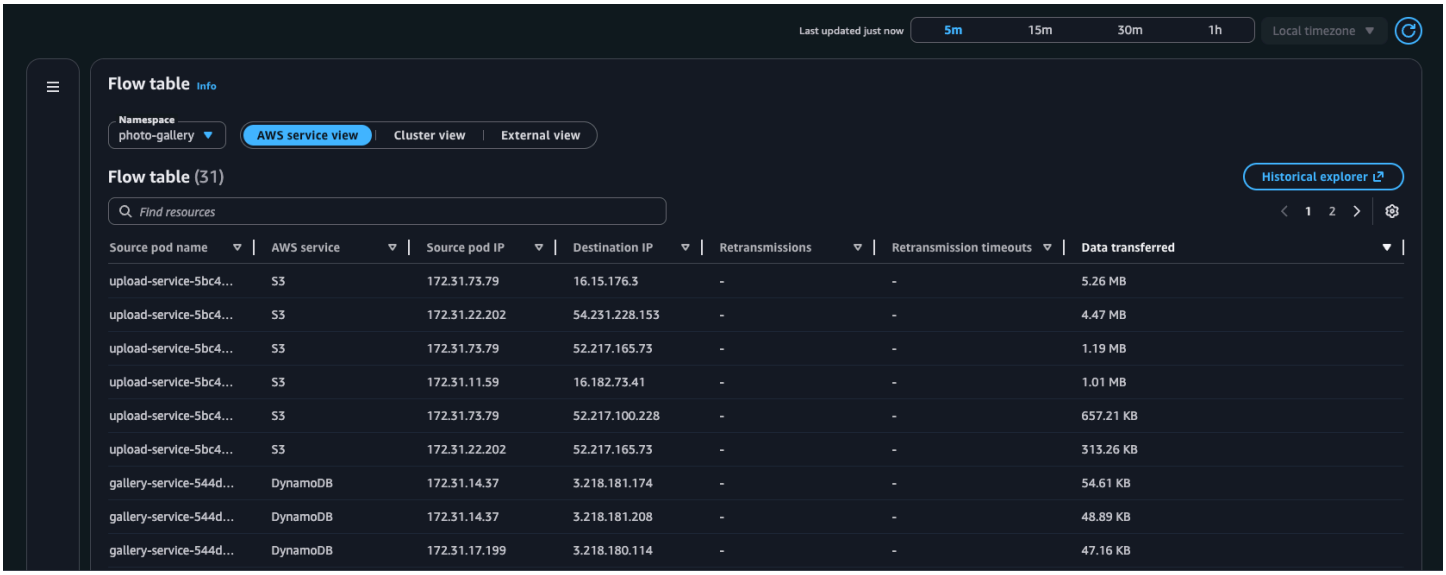



## Vista Pod

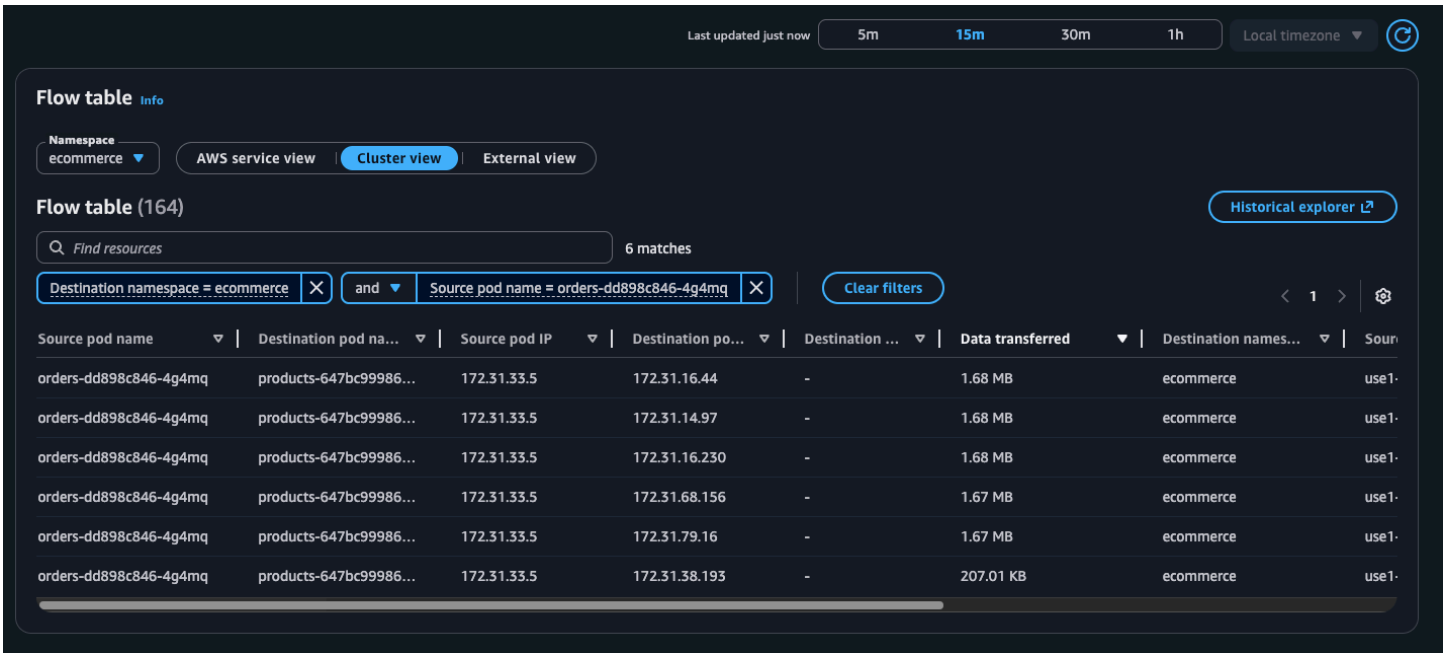



## Esempio: tabella di flusso

## AWS visualizzazione del servizio




Vista a grappolo




## Considerazioni e limitazioni

- L'osservabilità della rete di contenitori in EKS è disponibile solo nelle regioni in cui è supportato Network Flow Monitor.
- Le metriche di sistema supportate sono in OpenMetrics formato e possono essere estratte direttamente dall'agente Network Flow Monitor (NFM).

- Per abilitare l'osservabilità della rete dei container in EKS utilizzando Infrastructure as Code (IaC) come Terraform, è necessario definire e creare queste dipendenze nelle configurazioni: ambito NFM, monitor NFM e agente NFM.
- Network Flow Monitor supporta fino a circa 5 milioni di flussi al minuto. Si tratta di circa 5.000 istanze EC2 (nodi di lavoro EKS) con l'agente Network Flow Monitor installato. L'installazione di agenti su più di 5000 istanze può influire sulle prestazioni di monitoraggio fino a quando non sarà disponibile capacità aggiuntiva.
- È necessario eseguire una versione minima di 1.1.0 per il componente aggiuntivo EKS dell'agente NFM.
- È necessario utilizzare la versione 6.21.0 o successiva di Terraform AWS Provider per il supporto delle risorse di Network Flow Monitor.
- Per arricchire i flussi di rete con i metadati dei pod, i pod devono funzionare nel proprio spazio dei nomi di rete isolato, non nello spazio dei nomi della rete host.

# Monitoraggio delle metriche del cluster con Prometheus

Prometheus è un database di monitoraggio e di serie temporali che esegue lo scraping degli endpoint. Offre la possibilità di eseguire query, aggregare e archiviare i dati raccolti. È possibile utilizzarlo anche per gli avvisi e l'aggregazione degli avvisi. Questo argomento spiega come configurare Prometheus come un'opzione gestita oppure open source. Il monitoraggio dei parametri del piano di controllo di Amazon EKS è un caso d'uso comune.

Amazon Managed Service for Prometheus è un servizio di monitoraggio e avviso compatibile con Prometheus che semplifica il monitoraggio di applicazioni e infrastrutture containerizzate su larga scala. È un servizio completamente gestito che dimensiona automaticamente l'importazione, l'archiviazione, le query e gli avvisi dei parametri. Si integra anche con i servizi di sicurezza di AWS per consentire un accesso rapido e sicuro ai tuoi dati. È possibile utilizzare il linguaggio di query open source ProMQL per fare una query e creare avvisi relativi ai parametri. Inoltre, è possibile utilizzare l'alert manager nel servizio gestito da Amazon per Prometheus per configurare le regole di avviso per gli avvisi critici. Quindi, è possibile inviare questi avvisi critici come notifiche a un argomento Amazon SNS.

Esistono diverse opzioni per utilizzare Prometheus con Amazon EKS:

- È possibile attivare le metriche Prometheus quando si crea per la prima volta un cluster Amazon EKS o è possibile creare il proprio scraper Prometheus personalizzato per i cluster esistenti. Entrambe queste opzioni sono trattate in questo argomento.

- È possibile implementare Prometheus utilizzando Helm. Per ulteriori informazioni, consulta the section called "Implementare con Helm".
- È possibile visualizzare le metriche grezze del piano di controllo in formato Prometheus. Per ulteriori informazioni, consulta the section called "Piano di controllo".

## Passaggio 1: attivazione delle metriche Prometheus

> **Important**
>
> Le risorse del servizio gestito da Amazon per Prometheus non rientrano nel ciclo di vita del cluster e devono essere gestite indipendentemente dal cluster. Quando si elimina il cluster, assicurarsi di eliminare anche tutti gli scraper applicabili per bloccare i relativi costi. Per ulteriori informazioni, consulta Trova ed elimina scraper nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

Prometheus rileva e raccoglie i parametri dal cluster tramite un modello basato su pull chiamato scraping. Gli scraper sono configurati per raccogliere dati dall'infrastruttura del cluster e dalle applicazioni containerizzate. Quando attivi l'opzione di invio delle metriche Prometheus, il servizio gestito da Amazon per Prometheus fornisce uno scraper privo di agente completamente gestito.

Se non hai ancora creato il cluster, è possibile attivare l'opzione per inviare i le metriche a Prometheus al momento della creazione del cluster per la prima volta. Nella console Amazon EKS, questa opzione si trova nel passaggio Configura osservabilità della creazione di un nuovo cluster. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".

Se si dispone già di un cluster esistente, è possibile creare il proprio scraper Prometheus. Per eseguire questa operazione nella console Amazon EKS, accedi alla scheda Osservabilità del cluster e scegli il pulsante Aggiungi scraper. Se preferisci farlo con l'API AWS o AWS CLI, consulta Creare uno scraper nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

Le seguenti opzioni sono disponibili quando si crea lo scraper con la console Amazon EKS.

Alias scraper

(Facoltativo) Inserisci un alias univoco per lo scraper.

Destinazione

Scegli un workspace Amazon Managed Service for Prometheus. un'area di lavoro è uno spazio logico dedicato all'archiviazione e all'interrogazione dei parametri di Prometheus. Con questo workspace, sarai in grado di visualizzare i le metriche Prometheus di tutti gli account che vi hanno accesso. L'opzione Crea nuovo workspace consente ad Amazon EKS di creare un workspace per tuo conto utilizzando l'alias del workspace che fornisci. Con l'opzione Seleziona workspace esistente, puoi selezionare un workspace esistente da un elenco a discesa. Per ulteriori informazioni sui workspace, consulta Gestione di workspace nella Guida per l'utente di Amazon Managed Service for Prometheus.

Accesso al servizio

Questa sezione riassume le autorizzazioni concesse per l'invio delle metriche Prometheus:

- Consentire ad Amazon Managed Service for Prometheus di descrivere il cluster Amazon EKS sottoposto a scraping
- Consentire la scrittura remota nel workspace gestito da Amazon per Prometheus

Se `AmazonManagedScraperRole` esiste già, viene utilizzato dallo scraper. Scegli il link `AmazonManagedScraperRole` per visualizzare i dettagli dell'autorizzazione. Se `AmazonManagedScraperRole` non esiste ancora, scegli il link Visualizza dettagli di autorizzazione per vedere le autorizzazioni specifiche che concedi inviando le metriche Prometheus.

Sottoreti

Modificare le sottoreti che sono ereditate dallo scraper. Se è necessario aggiungere un'opzione di sottoreti disattivata, torna al passaggio di creazione del cluster Specifica la rete.

Configurazione dello scraper

Modifica la configurazione dello scraper in formato YAML secondo le esigenze. A tale scopo, utilizza il modulo o carica un file YAML sostitutivo. Per ulteriori informazioni, consulta Configurazione dello scraper nella Guida per l'utente di Amazon Managed Service for Prometheus.

Amazon Managed Service for Prometheus si riferisce allo scraper agentless creato insieme al cluster come raccoglitore gestito da AWS. Per ulteriori informazioni sugli agenti di raccolta gestiti da AWS, consulta Imposta metriche con raccoglitori gestiti da AWS nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

> **Important**
>
> - Se crei uno scraper Prometheus utilizzando AWS CLI o AWS API, è necessario modificarne la configurazione per fornire allo scraper le autorizzazioni interne al cluster. Per ulteriori informazioni, consulta Configurazione del cluster Amazon EKS nella Guida per l'utente di Amazon Managed Service for Prometheus.
> - Se disponi di uno scraper Prometheus creato prima dell'11 novembre 2024 che utilizza `aws-auth ConfigMap` anziché le voci di accesso, devi aggiornarlo per accedere a metriche aggiuntive dal piano di controllo del cluster Amazon EKS. Per la configurazione aggiornata, consulta Configurazione manuale di Amazon EKS per l'accesso allo scraper nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

## Passaggio 2: utilizzo delle metriche Prometheus

Per ulteriori informazioni su come utilizzare le metriche Prometheus dopo averli attivati per il cluster, consulta la Guida per l'utente del servizio gestito da Amazon per Prometheus.

## Passaggio 3: gestione degli scraper Prometheus

Per gestire gli scraper, scegli la scheda Osservabilità nella console Amazon EKS. Una tabella mostra un elenco di scraper per il cluster, incluse informazioni come l'ID, l'alias, lo stato e la data di creazione dello scraper. È possibile aggiungere altri scraper, modificarli, eliminarli o visualizzare ulteriori informazioni sugli scraper correnti.

Per visualizzare ulteriori dettagli sullo scraper, scegli un link relativo all'ID dello scraper. Ad esempio, è possibile visualizzare ARN, ambiente, ID dell'area di lavoro, ruolo IAM, configurazione e informazioni sulla rete. È possibile utilizzare l'ID scraper come input per operazioni dell'API del servizio gestito da Amazon per Prometheus come `DescribeScraper`, `UpdateScraper` e `DeleteScraper`. Per ulteriori informazioni sull'uso dell'API Prometheus, consulta Riferimento delle API del servizio gestito da Amazon per Prometheus.

## Implementare Prometheus utilizzando Helm

In alternativa all'utilizzo del Servizio gestito da Amazon per Prometheus, è possibile implementare Prometheus nel cluster con Helm. Se Helm è già stato installato, è possibile controllarne la versione con il comando `helm version`. Helm è un programma di gestione del pacchetto per cluster

Kubernetes. Per ulteriori informazioni su Helm e su come installarlo, consultare the section called "Implementazione di app con Helm".

Dopo aver configurato Helm per il cluster Amazon EKS, è possibile utilizzarlo per implementare Prometheus attraverso i seguenti passaggi.

1. Creare un namespace Prometheus.

```
kubectl create namespace prometheus
```

2. Aggiungere il repository del grafico `prometheus-community`.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Implementare Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> Se si verifica l'errore `Error: failed to download "stable/prometheus" (hint: running helm repo update may help)` durante l'esecuzione di questo comando`helm repo update prometheus-community`, esegui e riprova a eseguire il comando Passaggio 3.

Se si riceve l'errore `Error: rendered manifests contain a resource that already exists`, eseguire `helm uninstall` *`your-release-name`* `-n` *`namespace`* , quindi provare a eseguire nuovamente il comando della fase 3.

4. Verificare che tutti i pod nel namespace `prometheus` siano nello stato `READY`.

```
kubectl get pods -n prometheus
```

Di seguito viene riportato un output di esempio.

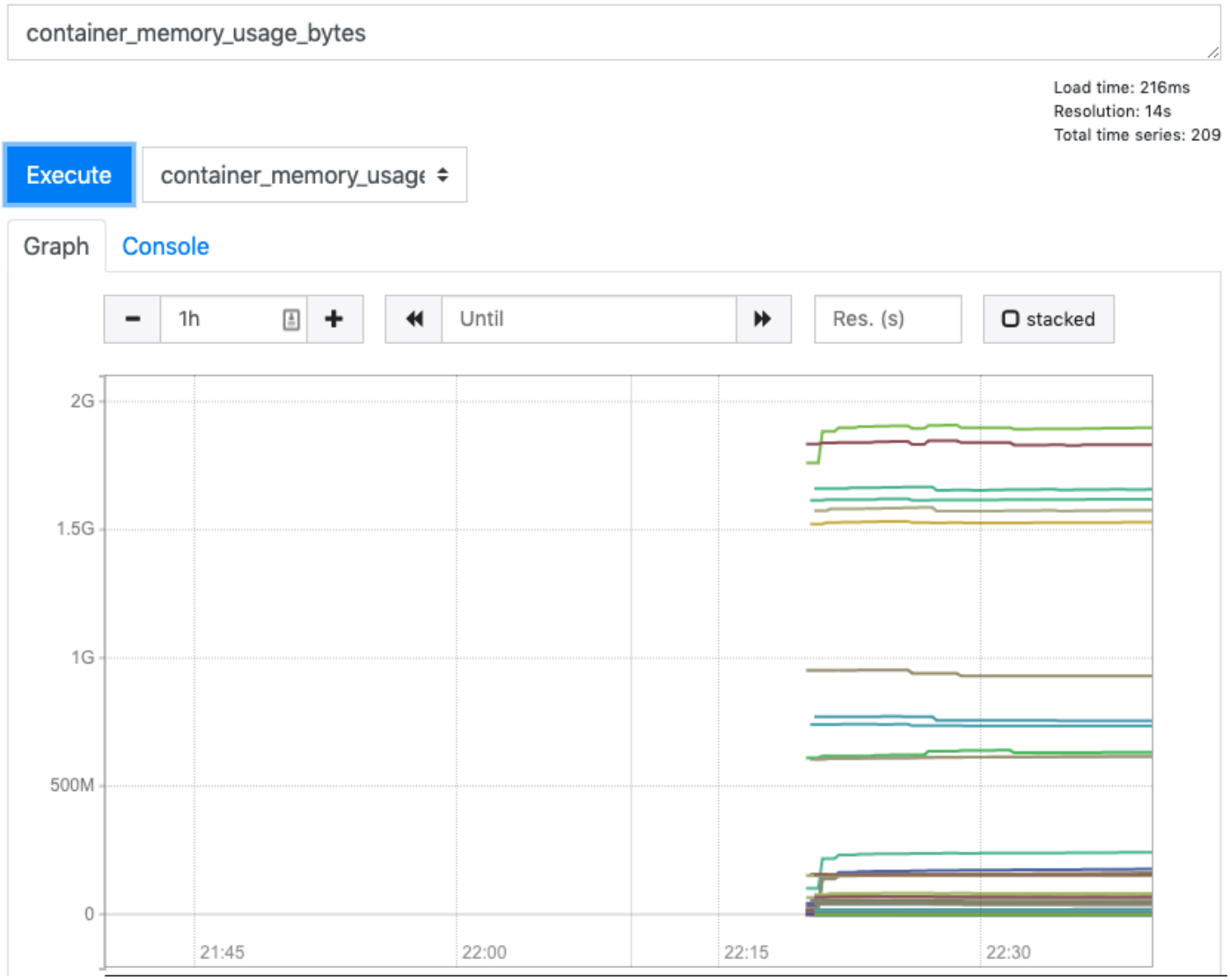
```
NAME                                             READY   STATUS    RESTARTS   AGE
```

```
prometheus-alertmanager-59b4c8c744-r7bgp          1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f    1/1     Running   0          48s
prometheus-node-exporter-jcjqz                    1/1     Running   0          48s
prometheus-node-exporter-jxv2h                    1/1     Running   0          48s
prometheus-node-exporter-vbdks                    1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw           1/1     Running   0          48s
prometheus-server-775957f748-mmht9                1/2     Running   0          48s
```

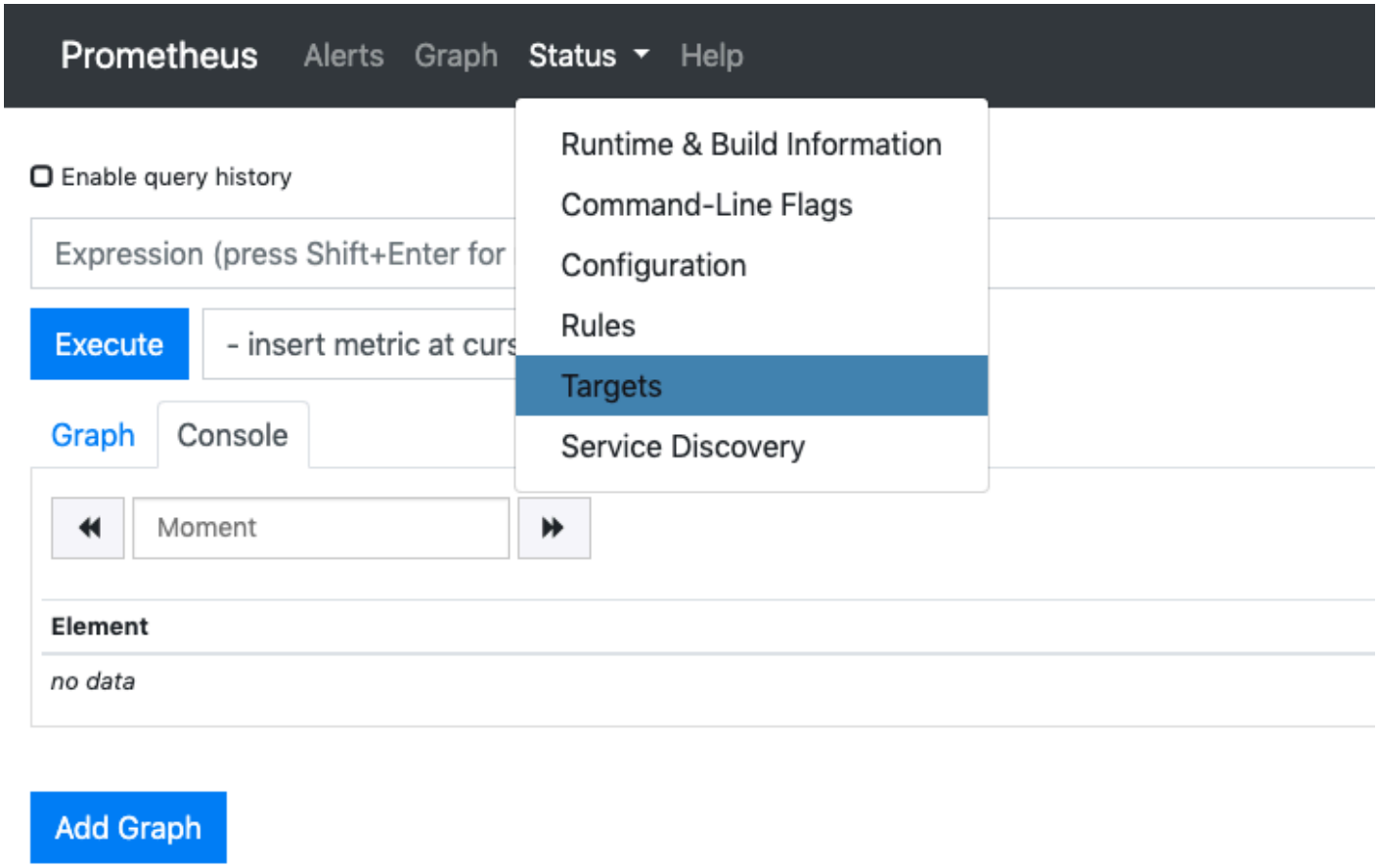
5. Utilizzare `kubectl` per eseguire l'inoltro della porta della console Prometheus al computer locale.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Indirizzare un browser web a `http://localhost:9090` per visualizzare la console Prometheus.
7. Scegliere un parametro dal menu - insert metric at cursor, quindi selezionare Execute (Esegui). Scegliere la scheda Graph (Grafico) per visualizzare il parametro nel tempo. L'immagine che segue mostra `container_memory_usage_bytes` nel tempo.




8. Dalla barra di navigazione superiore, scegliere Status (Stato), quindi Targets (Destinazioni).




Vengono visualizzati tutti gli endpoint Kubernetes che sono connessi a Prometheus utilizzando il rilevamento di servizi.

## Recupero delle metriche grezze del piano di controllo in formato Prometheus

Il piano di controllo Kubernetes espone una serie di metriche che sono rappresentate in un formato Prometheus. Questi parametri sono utili per il monitoraggio e l'analisi. Sono esposti internamente tramite endpoint metrici ed è possibile accedervi senza dover implementare completamente Prometheus. Tuttavia, l'implementazione più semplice di Prometheus consente di analizzare le metriche nel tempo.

Per visualizzare il risultato delle metriche grezze, sostituisci `endpoint` ed esegui il comando riportato.

```
kubectl get --raw endpoint
```

Questo comando consente di inoltrare qualsiasi endpoint e restituisce la risposta non elaborata. I diversi parametri sono elencati per riga e ogni riga include un nome di parametro, tag e un valore.

```
metric_name{tag="value"[,...]} value
```

## Recuperare le metriche dal server API

L'endpoint del server API generale è esposto sul piano di controllo Amazon EKS. Questo endpoint è utile principalmente per esaminare una metrica specifica.

```
kubectl get --raw /metrics
```

Di seguito viene riportato un output di esempio.

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

Questo output non elaborato restituisce in modo completo ciò che il server API espone.

## Recupera le metriche del piano di controllo con **`metrics.eks.amazonaws.com`**

Per i cluster con versione Kubernetes `1.28` e successive, Amazon EKS espone anche i parametri relativi al gruppo di API `metrics.eks.amazonaws.com`. Queste metriche includono componenti del piano di controllo come `kube-scheduler` e `kube-controller-manager`.

> **Note**
>
> Se disponi di una configurazione webhook che potrebbe bloccare la creazione della nuova risorsa `APIService` `v1.metrics.eks.amazonaws.com` nel cluster, la funzionalità degli endpoint delle metriche potrebbe non essere disponibile. È possibile verificarlo nel log di verifica `kube-apiserver` cercando la parola chiave `v1.metrics.eks.amazonaws.com`.

Recuperare le metriche **`kube-scheduler`**

Utilizza il codice seguente per recuperare le metriche `kube-scheduler`.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics"
```

Di seguito viene riportato un output di esempio.

```
# TYPE scheduler_pending_pods gauge
scheduler_pending_pods{queue="active"} 0
scheduler_pending_pods{queue="backoff"} 0
scheduler_pending_pods{queue="gated"} 0
scheduler_pending_pods{queue="unschedulable"} 18
# HELP scheduler_pod_scheduling_attempts [STABLE] Number of attempts to successfully
 schedule a pod.
# TYPE scheduler_pod_scheduling_attempts histogram
scheduler_pod_scheduling_attempts_bucket{le="1"} 79
scheduler_pod_scheduling_attempts_bucket{le="2"} 79
scheduler_pod_scheduling_attempts_bucket{le="4"} 79
scheduler_pod_scheduling_attempts_bucket{le="8"} 79
scheduler_pod_scheduling_attempts_bucket{le="16"} 79
scheduler_pod_scheduling_attempts_bucket{le="+Inf"} 81
[...]
```

A causa della quantità potenzialmente elevata di dati restituiti, esiste un endpoint di `kube-scheduler` metriche aggiuntivo per il recupero `kube_pod_resource_request` e le metriche. `kube_pod_resource_limit` Per recuperare queste metriche, usa il seguente comando.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/resourcemetrics"
```

Di seguito viene riportato un output di esempio.

```
# HELP kube_pod_resource_limit [STABLE] Resources limit for workloads on the cluster,
 broken down by pod. This shows the resource usage the scheduler and kubelet expect per
 pod for resources along with the unit for the resource if any.
# TYPE kube_pod_resource_limit gauge
kube_pod_resource_limit{namespace="kube-system",node="ip-192-168-87-3.us-
west-2.compute.internal",pod="coredns-7bf648ff5d-
dj4ss",priority="2000000000",resource="memory",scheduler="default-
scheduler",unit="bytes"} 1.7825792e+08
# HELP kube_pod_resource_request [STABLE] Resources requested by workloads on the
 cluster, broken down by pod. This shows the resource usage the scheduler and kubelet
 expect per pod for resources along with the unit for the resource if any.
# TYPE kube_pod_resource_request gauge
kube_pod_resource_request{namespace="kube-system",node="ip-192-168-87-3.us-
west-2.compute.internal",pod="aws-node-
x7znh",priority="2000001000",resource="cpu",scheduler="default-scheduler",unit="cores"}
 0.05
kube_pod_resource_request{namespace="kube-system",node="ip-192-168-87-3.us-
west-2.compute.internal",pod="coredns-7bf648ff5d-
dj4ss",priority="2000000000",resource="cpu",scheduler="default-scheduler",unit="cores"}
 0.1
[...]
```

Recuperare le metriche **`kube-controller-manager`**

Utilizza il codice seguente per recuperare le metriche `kube-controller-manager`.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics"
```

Di seguito viene riportato un output di esempio.

```
[...]
workqueue_work_duration_seconds_sum{name="pvprotection"} 0
workqueue_work_duration_seconds_count{name="pvprotection"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-08"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-07"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-05"} 19
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.001"} 109
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.01"} 139
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.1"} 181
workqueue_work_duration_seconds_bucket{name="replicaset",le="1"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="10"} 191
```

```
workqueue_work_duration_seconds_bucket{name="replicaset",le="+Inf"} 191
workqueue_work_duration_seconds_sum{name="replicaset"} 4.265655885000002
[...]
```

Comprendere le metriche di gestione del sistema di pianificazione e del controller

La tabella seguente descrive le metriche di gestione del sistema di pianificazione e del controller rese disponibili per lo scraping in stile Prometheus. Per ulteriori informazioni relative a queste metriche, consulta Riferimento delle metriche Kubernetes nella documentazione Kubernetes.

| Metrica | Componente del piano di controllo | Description |
|---|---|---|
| scheduler_pending_pods | pianificatore | Il numero di pod in attesa di essere pianificati su un nodo per l'esecuzione. |
| scheduler_schedule_attempts_total | pianificatore | Il numero di tentativi effettuati per pianificare i pod. |
| scheduler_preemption_attempts_total | pianificatore | Il numero di tentativi effettuati dal sistema di pianificazione per i pod con priorità più alta eliminando quelli con priorità inferiore. |
| scheduler_preemption_victims | pianificatore | Il numero di pod che sono stati selezionati per l'espulsione per fare spazio ai pod con priorità più alta. |
| scheduler_pod_scheduling_attempts | pianificatore | Il numero di tentativi di pianificare correttamente un pod. |
| scheduler_scheduling_attempt_duration_seconds | pianificatore | Indica la velocità o la lentezza con cui lo strumento di pianificazione riesce a trovare un luogo adatto per l'esecuzi |

| Metrica | Componente del piano di controllo | Description |
| --- | --- | --- |
| | | one di un pod in base a vari fattori come la disponibilità delle risorse e le regole di pianificazione. |
| scheduler_pod_scheduling_sli_duration_seconds | pianificatore | La latenza end-to-end per un pod in fase di pianificazione, dal momento in cui il pod entra nella coda di pianificazione. Ciò potrebbe comportare più tentativi di pianificazione. |
| kube_pod_resource_limit | pianificatore | Limite di risorse per i carichi di lavoro sul cluster, suddiviso per pod. Questo mostra l'utilizzo delle risorse che lo scheduler e kubelet prevedono per ogni pod per le risorse, insieme all'eventuale unità per la risorsa. |
| kube_pod_resource_request | pianificatore | Risorse richieste dai carichi di lavoro sul cluster, suddivise per pod. Questo mostra l'utilizzo delle risorse che lo scheduler e kubelet prevedono per ogni pod per le risorse, insieme all'eventuale unità per la risorsa. |
| cronjob_controller_job_creation_skew_duration_seconds | gestore del controller | Il tempo che intercorre tra la pianificazione dell'esecuzione di un cronjob e la creazione del processo corrispondente. |

| Metrica | Componente del piano di controllo | Description |
| --- | --- | --- |
| workqueue_depth | gestore del controller | La profondità corrente della coda. |
| workqueue_adds_total | gestore del controller | Il numero totale di aggiunte gestite dalla coda di lavoro. |
| workqueue_queue_duration_seconds | gestore del controller | Il tempo in secondi in cui un elemento rimane nella coda di lavoro prima di essere richiesto. |
| workqueue_work_duration_seconds | gestore del controller | Il tempo in secondi impiegato per l'elaborazione di un elemento dalla coda di lavoro. |

## Implementare uno scraper Prometheus per acquisire metriche in modo coerente

Per implementare uno scraper Prometheus per eseguire lo scraping delle metriche in modo coerente, utilizza la seguente configurazione:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-conf
data:
  prometheus.yml: |-
    global:
      scrape_interval: 30s
    scrape_configs:
    # apiserver metrics
    - job_name: apiserver-metrics
      kubernetes_sd_configs:
      - role: endpoints
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
```

```
      insecure_skip_verify: true
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    relabel_configs:
    - source_labels:
        [
          __meta_kubernetes_namespace,
          __meta_kubernetes_service_name,
          __meta_kubernetes_endpoint_port_name,
        ]
      action: keep
      regex: default;kubernetes;https
  # Scheduler metrics
  - job_name: 'ksh-metrics'
    kubernetes_sd_configs:
    - role: endpoints
    metrics_path: /apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics
    scheme: https
    tls_config:
      ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
      insecure_skip_verify: true
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    relabel_configs:
    - source_labels:
        [
          __meta_kubernetes_namespace,
          __meta_kubernetes_service_name,
          __meta_kubernetes_endpoint_port_name,
        ]
      action: keep
      regex: default;kubernetes;https
  # Controller Manager metrics
  - job_name: 'kcm-metrics'
    kubernetes_sd_configs:
    - role: endpoints
    metrics_path: /apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics
    scheme: https
    tls_config:
      ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
      insecure_skip_verify: true
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    relabel_configs:
    - source_labels:
        [
          __meta_kubernetes_namespace,
```

```
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
---
apiVersion: v1
kind: Pod
metadata:
  name: prom-pod
spec:
  containers:
  - name: prom-container
    image: prom/prometheus
    ports:
    - containerPort: 9090
    volumeMounts:
    - name: config-volume
      mountPath: /etc/prometheus/
  volumes:
  - name: config-volume
    configMap:
      name: prometheus-conf
```

L'autorizzazione che segue è richiesta per consentire al pod di accedere al nuovo endpoint metrico.

```
{
  "effect": "allow",
  "apiGroups": [
    "metrics.eks.amazonaws.com"
  ],
  "resources": [
    "kcm/metrics",
    "ksh/metrics"
  ],
  "verbs": [
    "get"
  ] },
```

Per applicare una patch al ruolo, puoi utilizzare il seguente comando.

```
kubectl patch clusterrole <role-name> --type=json -p='[
  {
```

```
    "op": "add",
    "path": "/rules/-",
    "value": {
      "verbs": ["get"],
      "apiGroups": ["metrics.eks.amazonaws.com"],
      "resources": ["kcm/metrics", "ksh/metrics"]
    }
  }
]'
```

Quindi puoi visualizzare la dashboard di Prometheus effettuando il proxy della porta dello scraper Prometheus verso la porta locale.

```
kubectl port-forward pods/prom-pod 9090:9090
```

Per il tuo cluster Amazon EKS, anche le principali metriche del piano di controllo Kubernetes vengono inserite in Amazon Metrics sotto lo spazio dei nomi. CloudWatch `AWS/EKS` Per visualizzarli, apri la CloudWatch console e seleziona Tutte le metriche dal riquadro di navigazione a sinistra. Nella pagina di selezione Metriche, scegli il namespace `AWS/EKS` e una dimensione delle metriche per il tuo cluster.

# Monitora i dati del cluster con Amazon CloudWatch

Amazon CloudWatch è un servizio di monitoraggio che raccoglie metriche e log dalle tue risorse cloud. CloudWatch fornisce gratuitamente alcuni parametri di base di Amazon EKS quando si utilizza un nuovo cluster di versione `1.28` o superiore. Tuttavia, quando si utilizza CloudWatch Observability Operator come componente aggiuntivo di Amazon EKS, è possibile ottenere funzionalità di osservabilità avanzate.

## Metriche di base in Amazon CloudWatch

Per i cluster con versione Kubernetes `1.28` e successive, ottieni metriche CloudWatch vendute gratuitamente nel namespace. `AWS/EKS` La seguente tabella fornisce un elenco delle metriche di base disponibili per le versioni supportate. Ogni metrica presente nell'elenco ha una frequenza di un minuto.

| Nome parametro | Description |
| --- | --- |
| `apiserver_flowcontrol_current_executing_seats` | Il numero di postazioni attualmente in uso per l'esecuzione delle richieste API. L'allocazione dei posti è determinata dalla configurazione priority_level e flow_schema nella funzionalità Priority and Fairness dell'API Kubernetes.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `scheduler_schedule_attempts_total` | Il numero totale di tentativi da parte pianificatore di pianificare i pod nel cluster per un determinato periodo. Questa metrica aiuta a monitorare il carico di lavoro del pianificatore e può indicare la pressione della pianificazione o potenziali problemi con il posizionamento dei pod.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `scheduler_schedule_attempts_SCHEDULED` | Il numero di tentativi riusciti da parte pianificatore di pianificare i pod sui nodi nel cluster per un determinato periodo.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `scheduler_schedule_attempts_UNSCHEDULABLE` | Il numero di tentativi di pianificazione dei pod che non è stato possibile pianificare per un determinato periodo a causa di vincoli validi, quali CPU o memoria insufficienti su un nodo.<br><br>Unità: numero |

| Nome parametro | Description |
| --- | --- |
| | Statistiche valide: somma |
| scheduler_schedule_attempts_ERROR | Il numero di tentativi di pianificazione dei pod che non sono riusciti in un determinato periodo a causa di un problema interno del pianificatore stesso, ad esempio problemi di connettività del server API.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| scheduler_pending_pods | Il numero totale di pod in sospeso che il pianificatore del cluster deve pianificare per un determinato periodo.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| scheduler_pending_pods_ACTIVEQ | Il numero di pod in sospeso in ActiveQ che attendono di essere pianificati nel cluster in un determinato periodo.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| scheduler_pending_pods_UNSCHEDULABLE | Il numero di pod in sospeso che il pianificatore ha tentato di pianificare senza successo e che vengono mantenuti in uno stato non pianificabile per un nuovo tentativo.<br><br>Unità: numero<br><br>Statistiche valide: somma |

| Nome parametro | Description |
| --- | --- |
| `scheduler_pending_pods_BACKOFF` | Il numero di Pod in sospeso in `backoffQ` in uno stato di backoff che sono in attesa della scadenza del periodo di backoff.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `scheduler_pending_pods_GATED` | Il numero di pod in sospeso che sono attualmente in attesa in uno stato chiuso, in quanto non possono essere pianificati finché non soddisfano le condizioni richieste.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_request_total` | Il numero di richieste HTTP inviate su tutti i server API del cluster.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_request_total_4XX` | Il numero di richieste HTTP effettuate a tutti i server API del cluster che hanno generato codici di stato `4XX` (errore client).<br><br>Unità: numero<br><br>Statistiche valide: somma |

| Nome parametro | Description |
| --- | --- |
| apiserver_request_total_429 | Il numero di richieste HTTP inviate a tutti i server API del cluster che hanno generato il codice di stato 429, che si verifica quando i client superano le soglie di limitazione della velocità.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| apiserver_request_total_5XX | Il numero di richieste HTTP effettuate a tutti i server API del cluster che hanno generato codici di stato 5XX (errore server).<br><br>Unità: numero<br><br>Statistiche valide: somma |
| apiserver_request_total_LIST_PODS | Il numero di richieste Pod LIST inviate a tutti i server API del cluster.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| apiserver_request_duration_seconds_PUT_P99 | Il 99° percentile della latenza per le richieste PUT calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste PUT.<br><br>Unità: secondi<br><br>Statistiche valide: media |

| Nome parametro | Description |
| --- | --- |
| `apiserver_request_duration_seconds_PATCH_P99` | Il 99° percentile della latenza per le richieste `PATCH` calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste `PATCH`.<br><br>Unità: secondi<br><br>Statistiche valide: media |
| `apiserver_request_duration_seconds_POST_P99` | Il 99° percentile della latenza per le richieste `POST` calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste `POST`.<br><br>Unità: secondi<br><br>Statistiche valide: media |
| `apiserver_request_duration_seconds_GET_P99` | Il 99° percentile della latenza per le richieste `GET` calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste `GET`.<br><br>Unità: secondi<br><br>Statistiche valide: media |

| Nome parametro | Description |
| --- | --- |
| `apiserver_request_duration_seconds_LIST_P99` | Il 99° percentile della latenza per le richieste `LIST` calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste `LIST`.<br><br>Unità: secondi<br><br>Statistiche valide: media |
| `apiserver_request_duration_seconds_DELETE_P99` | Il 99° percentile della latenza per le richieste `DELETE` calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste `DELETE`.<br><br>Unità: secondi<br><br>Statistiche valide: media |
| `apiserver_current_inflight_requests_MUTATING` | Il numero di richieste di mutazione (`POST`, `PUT`, `DELETE`, `PATCH`) attualmente in elaborazione su tutti i server API del cluster. Questa metrica rappresenta le richieste in corso per le quali l'elaborazione non è ancora stata completata.<br><br>Unità: numero<br><br>Statistiche valide: somma |

| Nome parametro | Description |
| --- | --- |
| `apiserver_current_inflight_requests_READONLY` | Il numero di richieste di sola lettura (`GET`, `LIST`) attualmente in elaborazione su tutti i server API del cluster. Questa metrica rappresenta le richieste in corso per le quali l'elaborazione non è ancora stata completata.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_admission_webhook_request_total` | Il numero di richieste webhook di ammissione inviate su tutti i server API nel cluster.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_admission_webhook_request_total_ADMIT` | Il numero di richieste di mutazione webhook di ammissione inviate su tutti i server API nel cluster.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_admission_webhook_request_total_VALIDATING` | Il numero di richieste webhook di ammission e di convalida inviate su tutti i server API nel cluster.<br><br>Unità: numero<br><br>Statistiche valide: somma |

| Nome parametro | Description |
| --- | --- |
| `apiserver_admission_webhook_rejection_count` | Il numero di richieste webhook di ammissione inviate su tutti i server API del cluster che sono state rifiutate.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_admission_webhook_rejection_count_ADMIT` | Il numero di richieste di mutazione webhook di ammissione inviate su tutti i server API del cluster che sono state rifiutate.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_admission_webhook_rejection_count_VALIDATING` | Il numero di richieste webhook di ammission e di convalida inviate su tutti i server API del cluster che sono state rifiutate.<br><br>Unità: numero<br><br>Statistiche valide: somma |
| `apiserver_admission_webhook_admission_duration_seconds` | Il 99° percentile della latenza per le richieste webhook di ammissione di terze parti calcolato sulla base di tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste webhook di ammission e di terze parti.<br><br>Unità: secondi<br><br>Statistiche valide: media |

| Nome parametro | Description |
| --- | --- |
| apiserver_admission_webhook_admission_duration_seconds_ADMIT_P99 | Il 99° percentile della latenza per le richieste webhook di mutazione di ammissione di terze parti calcolato sulla base di tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste webhook di mutazione di ammissione di terze parti.<br><br>Unità: secondi<br><br>Statistiche valide: media |
| apiserver_admission_webhook_admission_duration_seconds_VALIDATING_P99 | Il 99° percentile della latenza per le richieste webhook di validazione di ammissione di terze parti calcolato sulla base di tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste webhook di validazione di ammissione di terze parti.<br><br>Unità: secondi<br><br>Statistiche valide: media |
| etcd_mvcc_db_total_size_in_bytes | Dimensione totale del database sottostante allocata fisicamente in byte. Si noti che questa metrica misura l'allocazione fisica del disco e non viene utilizzata per l'applicazione delle quote. (noto anche come apiserver_storage_size_bytes)<br><br>Unità: byte<br><br>Statistiche valide: massimo |

| Nome parametro | Description |
| --- | --- |
| `etcd_mvcc_db_total_size_in_use_in_bytes` | La dimensione effettiva dei dati del database etcd, escluso lo spazio libero in attesa della deframmentazione. Questa metrica indica l'utilizzo corrente del database e determina se il cluster supererà la quota di dimensioni del database e entrerà in modalità di sola lettura.<br><br>Unità: byte<br><br>Statistiche valide: massimo |

## Operatore di CloudWatch osservabilità di Amazon

Amazon CloudWatch Observability raccoglie log, parametri e dati di tracciamento in tempo reale. Li invia ad Amazon CloudWatch e AWS X-Ray. Puoi installare questo componente aggiuntivo per abilitare sia CloudWatch Application Signals che CloudWatch Container Insights con una migliore osservabilità per Amazon EKS. In questo modo puoi monitorare lo stato e le prestazioni dell'infrastruttura e delle applicazioni containerizzate. Amazon CloudWatch Observability Operator è progettato per installare e configurare i componenti necessari.

Amazon EKS supporta CloudWatch Observability Operator come componente aggiuntivo di Amazon EKS. Il componente aggiuntivo consente l'uso di Container Insights sui nodi di lavoro Linux e Windows nel cluster. Per abilitare Container Insights su Windows, la versione del componente aggiuntivo Amazon EKS deve essere `1.5.0` o superiore. Attualmente, CloudWatch Application Signals non è supportato su Amazon EKS Windows.

Gli argomenti seguenti descrivono come iniziare a utilizzare CloudWatch Observability Operator per il tuo cluster Amazon EKS.

- Per istruzioni sull'installazione di questo componente aggiuntivo, consulta Installa l' CloudWatch agente con il componente aggiuntivo Amazon CloudWatch Observability EKS o il grafico Helm nella Amazon User Guide. CloudWatch
- Per ulteriori informazioni su CloudWatch Application Signals, consulta Application Signals nella Amazon CloudWatch User Guide.

- Per ulteriori informazioni su Container Insights, consulta Using Container Insights nella Amazon CloudWatch User Guide.

## Invia i log del piano di controllo a CloudWatch Logs

La registrazione del piano di controllo di Amazon EKS fornisce log di audit e diagnostica direttamente dal piano di controllo di Amazon EKS ai CloudWatch log del tuo account. Questi log consentono di semplificare la protezione e l'esecuzione dei cluster. Puoi selezionare i tipi di log esatti di cui hai bisogno e i log vengono inviati come flussi di log a un gruppo per ogni cluster Amazon EKS in. CloudWatch Puoi utilizzare i filtri di CloudWatch abbonamento per eseguire analisi in tempo reale sui log o per inoltrarli ad altri servizi (i log saranno codificati in Base64 e compressi con il formato gzip). Per ulteriori informazioni, consulta Amazon logging. CloudWatch

> **Note**
>
> I log del piano di controllo di Amazon EKS vengono inviati a CloudWatch Logs in pochi minuti. Tuttavia, la consegna dei log è la soluzione migliore.

É possibile iniziare a utilizzare il piano di controllo Amazon EKS scegliendo quali tipi di log che si desidera abilitare per ogni cluster Amazon EKS nuovo o esistente. Puoi abilitare o disabilitare ogni tipo di log per cluster utilizzando l'interfaccia a riga di Console di gestione AWS comando AWS (versione `1.16.139` o successiva) o tramite l'API Amazon EKS. Se abilitata, i log vengono inviati automaticamente dal cluster Amazon EKS a CloudWatch Logs nello stesso account.

Quando si utilizza la registrazione del piano di controllo Amazon EKS, viene addebitato il prezzo Amazon EKS standard per ogni cluster eseguito. Ti vengono addebitati i costi standard CloudWatch di inserimento e archiviazione dei dati di Logs per tutti i log inviati a CloudWatch Logs dai tuoi cluster. Ti vengono inoltre addebitate tutte AWS le risorse, come le istanze Amazon EC2 o i volumi Amazon EBS, di cui esegui il provisioning come parte del tuo cluster.

Sono disponibili i seguenti tipi di registro del piano di controllo del cluster. Ogni tipo di registro corrisponde a un componente del piano di controllo Kubernetes. Per ulteriori informazioni su questi componenti, consultare Kubernetes Components nella documentazione Kubernetes.

Server API**`api`** ()

Il server API del cluster è il componente del piano di controllo (control-plane) che espone l'API Kubernetes. Se abiliti i log del server API quando avvii il cluster o poco dopo, questi log includono i flag del server API utilizzati per l'avvio del server API. Per ulteriori informazioni, consulta kube-apiserver e la policy di audit nella documentazione Kubernetes.

Audit**`audit`** ()

I log di audit Kubernetes forniscono un record dei singoli utenti, amministratori o componenti di sistema che interessano il cluster. Per ulteriori informazioni, consultare Auditing nella documentazione Kubernetes.

Autenticatore**`authenticator`** ()

I log dell'autenticatore sono univoci per Amazon EKS. Questi registri rappresentano il componente del piano di controllo utilizzato da Amazon EKS per l'autenticazione Role Based Access Control (RBAC) Kubernetes utilizzando credenziali IAM. Per ulteriori informazioni, consulta Gestione dei cluster.

Gestore controller**`controllerManager`** ()

Il gestore controller gestisce i loop di controllo di base inviati con Kubernetes. Per ulteriori informazioni, consultare kube-controller-manager nella documentazione Kubernetes.

Pianificatore**`scheduler`** ()

Il componente pianificatore gestisce quando e dove eseguire i pod nel cluster. Per ulteriori informazioni, consultare kube-scheduler nella documentazione Kubernetes.

## Abilitare o disabilitare i log del piano di controllo (control-plane)

Per impostazione predefinita, i log del piano di controllo del cluster non vengono inviati a Logs. CloudWatch È necessario abilitare ogni tipo di log singolarmente per inviare i log per il cluster. CloudWatch Le velocità di acquisizione dei log, archiviazione degli archivi e scansione dei dati si applicano ai log del piano di controllo abilitati. Per ulteriori informazioni, consulta Prezzi di CloudWatch .

Per aggiornare la configurazione di registrazione del piano di controllo (control-plane), Amazon EKS richiede fino a cinque indirizzi IP disponibili in ciascuna sottorete. Quando abiliti un tipo di registro, i log vengono inviati con un livello di dettaglio dei log di 2.

È possibile abilitare o disabilitare i log del piano di controllo (control-plane) con la Console di gestione AWS o AWS CLI.

## Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegli la scheda Osservabilità.
4. Nella sezione Registrazione del piano di controllo, scegli Gestisci registrazione.
5. Per ogni singolo tipo di log, scegli se il tipo di log deve essere attivato o disattivato. Per impostazione predefinita, i tipi di log sono disattivati.
6. Scegliere Salva le modifiche per terminare.

## AWS CLI

1. Controlla la tua versione AWS CLI con il seguente comando.

   ```
   aws --version
   ```

   Se la tua versione AWS CLI è precedente a `1.16.139`, devi prima aggiornarla alla versione più recente. Per installare o aggiornare la AWS CLI, vedere Installazione dell'interfaccia a riga di AWS comando nella Guida per l'utente dell'interfaccia a riga di AWS comando.
2. Aggiorna la configurazione di esportazione dei log del piano di controllo del cluster con il seguente comando AWS CLI. Sostituisci *`my-cluster`* con il nome del cluster e specifica i valori desiderati per l'accesso all'endpoint.

   > **Note**
   >
   > Il comando seguente invia tutti i tipi di log disponibili a CloudWatch Logs.

   ```
   aws eks update-cluster-config \
       --region region-code \
       --name my-cluster \
       --logging '{"clusterLogging":[{"types":
   ["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
   ```

Di seguito viene riportato un output di esempio.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

3. Monitorare lo stato di aggiornamento della configurazione del log con il comando seguente utilizzando il nome del cluster e l'ID di aggiornamento restituiti dal comando precedente. L'aggiornamento è completo quando lo stato viene visualizzato come `Successful`.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

Di seguito viene riportato un output di esempio.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
```

```
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

## Visualizzare i log del piano di controllo (control-plane) del cluster

Dopo aver abilitato uno qualsiasi dei tipi di log del piano di controllo per il tuo cluster Amazon EKS, puoi visualizzarli sulla CloudWatch console.

Per ulteriori informazioni sulla visualizzazione, l'analisi e la gestione dei log in CloudWatch, consulta la Amazon CloudWatch Logs User Guide.

1. Apri la CloudWatch console. Questo link apre la console e mostra i gruppi di log disponibili correnti e li filtra con il prefisso `/aws/eks`.
2. Scegliere il cluster per il quale si intende visualizzare i log. Il formato del nome del gruppo di log è `/aws/eks/`*`my-cluster`*`/cluster`.
3. Scegliere il flusso di log da visualizzare. L'elenco seguente descrive il formato del nome del flusso di log per ogni tipo di registro.

> **Note**
>
> Al crescere della quantità di dati del flusso di log, i nomi del flusso di log vengono ruotati. Quando esistono più flussi di log per un determinato tipo di log, puoi visualizzare l'ultimo flusso di log cercando il nome del flusso di log con Last event time (Ora ultimo evento) più recente.

- Log del componente server API Kubernetes (**`api`**) – `kube-apiserver-`*`1234567890abcdef01234567890abcde`*
- Audit (**`audit`**) – `kube-apiserver-audit-`*`1234567890abcdef01234567890abcde`*
- Autenticatore (**`authenticator`**) – `authenticator-`*`1234567890abcdef01234567890abcde`*
- Gestore controller (**`controllerManager`**) – `kube-controller-manager-`*`1234567890abcdef01234567890abcde`*

- Pianificatore (**`scheduler`**) – `kube-scheduler-`*`1234567890abcdef01234567890abcde`*

4. Esamina gli eventi del flusso di log.

   Ad esempio, dovresti vedere i flag iniziali del server API del cluster quando visualizzi la parte iniziale di `kube-apiserver-`*`1234567890abcdef01234567890abcde`* .

   > **Note**
   >
   > Se i log del server API non vengono visualizzati all'inizio del flusso di log, è probabile che il file di log del server API sia stato ruotato sul server prima di abilitare la registrazione di log del server API sul server. Tutti i file di log che vengono ruotati prima che la registrazione del server API sia abilitata non possono essere esportati in. CloudWatch

Tuttavia, è possibile creare un nuovo cluster con la stessa versione di Kubernetes e abilitare la registrazione del server API quando si crea il cluster. I cluster con la stessa versione della piattaforma hanno gli stessi flag abilitati, quindi i flag devono corrispondere a quelli del nuovo cluster. Quando finisci di visualizzare i flag per il nuovo cluster in CloudWatch, puoi eliminare il nuovo cluster.

# Registra le chiamate API come eventi AWS CloudTrail

Amazon EKS è integrato con AWS CloudTrail. CloudTrail è un servizio che fornisce un registro delle operazioni eseguite da un utente, un ruolo o un servizio AWS in Amazon EKS. CloudTrail acquisisce tutte le chiamate API per Amazon EKS come eventi. Questi includono le chiamate della console Amazon EKS e le chiamate del codice alle operazioni API Amazon EKS.

Se crei un percorso, puoi abilitare la distribuzione continua di eventi CloudTrail in un bucket Amazon S3, inclusi gli eventi per Amazon EKS. Se non configuri un trail, puoi comunque visualizzare gli eventi più recenti nella console di CloudTrail in Cronologia eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare diversi dettagli su una richiesta. Ad esempio, puoi determinare quando è stata effettuata una richiesta ad Amazon EKS, l'indirizzo IP da cui è stata eseguita e l'autore.

Per ulteriori informazioni su CloudTrail, consulta AWS CloudTrail User Guide.

Argomenti

- Visualizzazione dei riferimenti utili per AWS CloudTrail
- Analisi AWS CloudTrail voci dei file di registro

- Visualizzare i parametri per i gruppi Amazon EC2 Auto Scaling

## Visualizzazione dei riferimenti utili per AWS CloudTrail

Al momento della creazione del tuo account AWS, CloudTrail è abilitato sul tuo account AWS. Quando si verifica un'attività in Amazon EKS, questa viene registrata in un evento CloudTrail insieme ad altri eventi di servizio AWS nella Cronologia eventi. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi mediante la cronologia eventi di CloudTrail.

Per una registrazione continua degli eventi nell'account AWS, inclusi gli eventi per Amazon EKS, creare un percorso. Un percorso consente a CloudTrail di distribuire i file di log in un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le Regioni AWS. Il percorso registra gli eventi da tutte le regioni AWSnella partizione AWS e distribuisce i file di log nel bucket Simple Storage Service (Amazon S3) specificato. Inoltre, è possibile configurare altri servizi AWS per analizzare con maggiore dettaglio e utilizzare i dati evento raccolti nei log CloudTrail. Per ulteriori informazioni, consulta le risorse seguenti.

- Panoramica della creazione di un percorso
- Servizi e integrazioni CloudTrail supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di registro CloudTrail da più Regioni e Ricezione di file di log CloudTrail da più account

Tutte le operazioni Amazon EKS vengono registrate da CloudTrail e sono documentate nella Documentazione di riferimento delle API di Amazon EKS. Ad esempio, le chiamate alle sezioni CreateCluster, ListClusters e DeleteCluster generano voci nei file di log CloudTrail.

Ogni evento o voce di log include informazioni sul tipo di identità IAM che ha effettuato la richiesta e sulle credenziali utilizzate. Se sono state utilizzate credenziali temporanee, la voce mostra il modo in cui tali credenziali sono state ottenute.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

## Analisi AWS CloudTrail voci dei file di registro

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro.

Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, Ciò include informazioni come la data e l'ora dell'azione e i parametri di richiesta utilizzati. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'CreateClusterazione.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws: iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws: eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
```

```
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

## Voci di registro per ruoli collegati ai servizi Amazon EKS

I ruoli collegati al servizio Amazon EKS effettuano chiamate API alle AWS risorse. CloudTrail registra le voci `username: AWSServiceRoleForAmazonEKS` e le `username: AWSServiceRoleForAmazonEKSNodegroup` visualizza per le chiamate effettuate dai ruoli collegati al servizio Amazon EKS. Per ulteriori informazioni su Amazon EKS e sui ruoli collegati ai servizi, consultare the section called "Ruoli collegati ai servizi".

L'esempio seguente mostra una voce di CloudTrail registro che mostra un'`DeleteInstanceProfile`azione eseguita dal ruolo collegato al `AWSServiceRoleForAmazonEKSNodegroup` servizio, indicato in. `sessionContext`

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws: sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws: iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


## Visualizzare i parametri per i gruppi Amazon EC2 Auto Scaling

Nei gruppi di nodi gestiti da Amazon EKS i parametri dei gruppi di Amazon EC2 Auto Scaling sono abilitati per impostazione predefinita senza costi aggiuntivi. Il gruppo Auto Scaling invia dati campionati ad Amazon CloudWatch ogni minuto. Questi parametri possono essere perfezionati in base al nome dei gruppi Auto Scaling. Con questi parametri, si ottiene una visibilità quasi continua nella cronologia del gruppo Auto Scaling che alimenta i gruppi di nodi gestiti, ad esempio le modifiche delle dimensioni del gruppo nel tempo. I parametri di un gruppo Auto Scaling sono disponibili nella console di Amazon CloudWatch o in quella di Auto Scaling. Per ulteriori informazioni, consultare la pagina Monitor CloudWatch metrics for your Auto Scaling groups and instances.

Abilitando la raccolta dei parametri di un gruppo Auto Scaling, sarà possibile monitorare la scalabilità dei gruppi di nodi gestiti. I parametri di un gruppo con scalabilità automatica riportano la dimensione minima, massima e desiderata di un gruppo con scalabilità automatica. Puoi creare un allarme se il numero di nodi in un gruppo di nodi scende al di sotto della dimensione minima, fatto che indicherebbe un gruppo di nodi non integro. Il monitoraggio delle dimensioni dei gruppi di nodi è utile anche per regolare il numero massimo in modo che il piano dati non esaurisca la capacità.

Se si preferisce non raccogliere questi parametri, è possibile disabilitarli tutti o solo alcuni. Ad esempio, è possibile farlo per evitare il rumore nelle dashboard CloudWatch. Per ulteriori informazioni, consultare la pagina Amazon CloudWatch metrics for Amazon EC2 Auto Scaling.

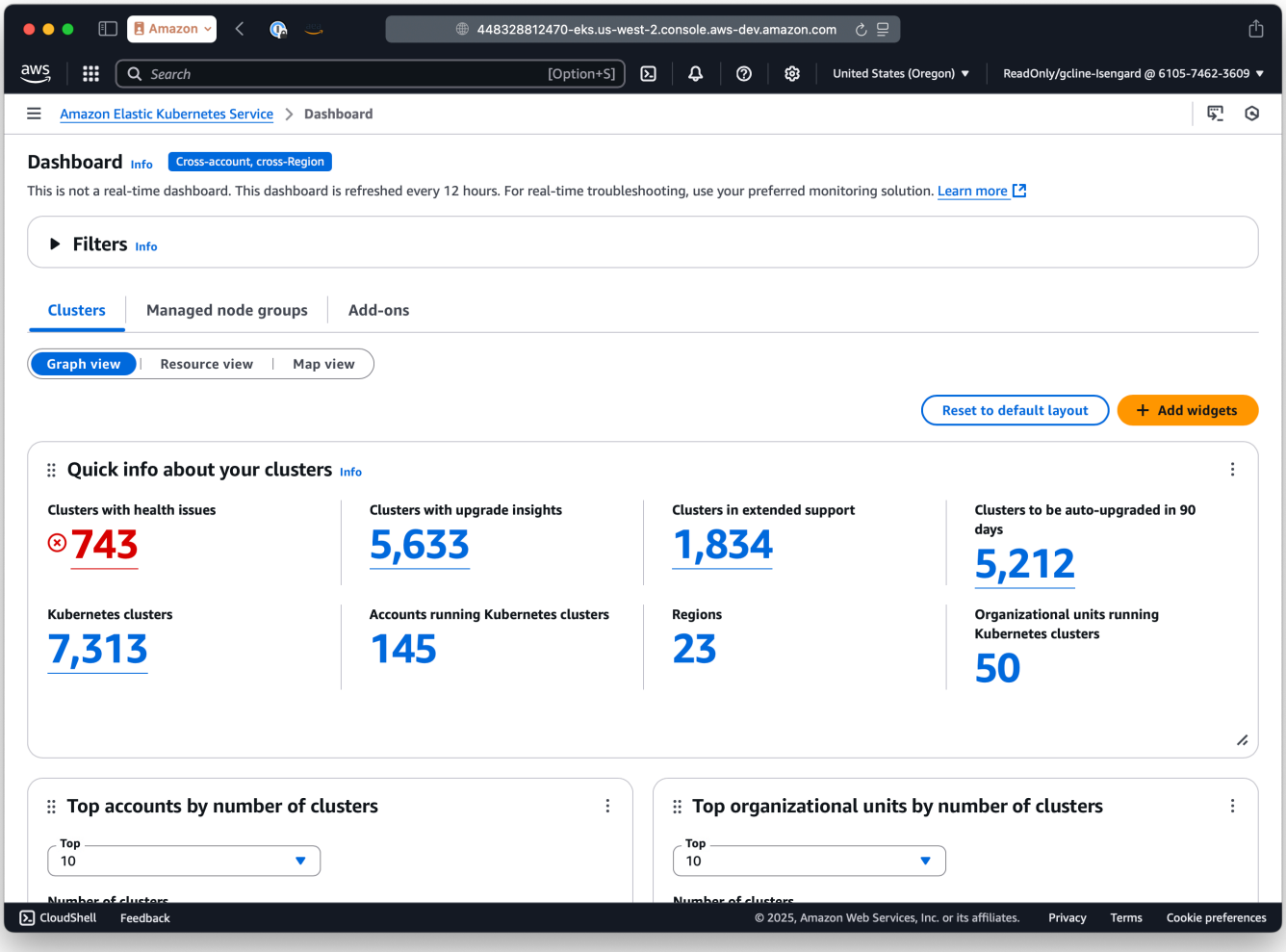
## Invio di dati metrici e di tracciamento con l'operatore ADOT

Amazon EKS supporta ora l'utilizzo di Console di gestione AWS, AWS CLI e l'API di Amazon EKS per installare e gestire l'operatore AWS Distro for OpenTelemetry (ADOT). Ciò rende più semplice la possibilità per le applicazioni in esecuzione su Amazon EKS di inviare parametri e dati di traccia a vari servizi di monitoraggio, ad esempio Amazon CloudWatch, Prometheus e X-Ray.

Per ulteriori informazioni, consulta Getting Started with AWS Distro for OpenTelemetry using EKS Add-Ons nella documentazione di AWS Distro per OpenTelemetry.

# Visualizza i dati aggregati sulle risorse del cluster con il pannello di controllo EKS

## Cos'è il pannello di controllo Amazon EKS?




Amazon EKS Dashboard offre una visibilità consolidata sui cluster Kubernetes su più regioni e account. AWS AWS Con questo pannello di controllo puoi:

- Tenere traccia dei cluster pianificati per gli aggiornamenti automatici di fine supporto entro i successivi 90 giorni.
- Progettare i costi del piano di controllo EKS per i cluster con supporto esteso.
- Esaminare i cluster con approfondimenti che richiedono attenzione prima dell'aggiornamento.
- Identificare i gruppi di nodi gestiti che eseguono versioni AMI specifiche.

- Monitorare la distribuzione del tipo di supporto del cluster (standard rispetto al supporto esteso).

Il pannello di controllo EKS si integra con EKS Cluster Insights per far emergere problemi con i cluster, come l'uso di API Kubernetes obsolete. Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

> **Note**
>
> Il pannello di controllo EKS non è in tempo reale e si aggiorna ogni 12 ore. Per il monitoraggio dei cluster in tempo reale, consulta. Monitoraggio dei cluster

## Come viene utilizzata la dashboard AWS Organizzazioni?

La dashboard di Amazon EKS richiede l'integrazione AWS delle organizzazioni per la funzionalità. Sfrutta le AWS organizzazioni per raccogliere in modo sicuro informazioni sui cluster tra gli account. Questa integrazione fornisce gestione e governance centralizzate man mano che l'infrastruttura AWS si espande.

Se AWS Organizations non è abilitato per la tua infrastruttura, consulta la Guida per l'utente AWS delle organizzazioni per le istruzioni di configurazione.

### Cross-region e accesso tra più account

L'EKS Dashboard può visualizzare le risorse del cluster in qualsiasi account membro dell' AWS organizzazione. Per generare un elenco di AWS account nella tua organizzazione, vedi Esportazione dei dettagli per tutti gli account di un'organizzazione.

La AWS regione us-east-1 genera il dashboard. Per visualizzare il pannello di controllo, devi accedere a questa regione. La dashboard aggrega i dati tra AWS le regioni, ma non include `GovCloud` le regioni della Cina.

### Termini chiave

- AWS Organizzazione: una struttura di gestione unificata per più AWS account.
- Account di gestione: l'account principale che controlla l' AWS organizzazione.
- Account membro: qualsiasi account all'interno dell'organizzazione ad eccezione dell'account di gestione.

- Amministratore delegato: account membro a cui sono state concesse autorizzazioni amministrative specifiche per più account. All'interno dell'account di gestione, è possibile selezionare un account amministratore delegato per AWS servizio.
- Accesso attendibile: autorizzazione per il pannello di controllo EKS ad accedere alle informazioni del cluster tra gli account dell'organizzazione.
- Service-Linked Ruolo (SLR): un tipo unico di ruolo IAM collegato direttamente a un AWS servizio. L'EKS Dashboard utilizza una SLR per leggere le informazioni sui tuoi account e sulle tue organizzazioni.
- Per ulteriori informazioni, vedere Terminologia e concetti nella Guida per l'utente AWS delle organizzazioni.

### Panoramica generale

1. Accedi all'account di gestione della tua AWS organizzazione.
   - I passaggi per accedere all'account di gestione dipendono da come hai configurato la tua AWS organizzazione. Ad esempio, puoi accedere all'account di gestione tramite AWS Identity Center o Okta.
2. Abilita l'accesso attendibile tramite la console EKS.
3. Assegna un amministratore delegato utilizzando il suo ID AWS account.
4. Passa all'account amministratore delegato.
5. Accedi alla console EKS migliorata con visibilità a livello di organizzazione.

## Abilita l'EKS Dashboard utilizzando il AWS console

> **Important**
>
> È necessario accedere all'account di gestione della propria AWS organizzazione per abilitare l'EKS Dashboard.

### Accedi alle impostazioni del pannello di controllo EKS

1. Conferma quanto segue:
   a. AWS Le organizzazioni sono abilitate e configurate.

b. Hai effettuato l'accesso all'account di gestione dell'organizzazione.

c. Stai visualizzando la console AWS di gestione nella regione us-east-1.

2. Passa alla console EKS.
3. Nella barra laterale sinistra, apri Impostazioni del pannello di controllo.

### Configurazione dell'accesso al pannello di controllo Amazon EKS

1. Trova l'ID dell' AWS account a cui desideri consentire la AWS visualizzazione del dashboard EKS.
   a. Questo passaggio è facoltativo ma consigliato. In caso contrario, puoi accedere al pannello di controllo solo dall'account di gestione. Come best practice, dovresti limitare gli accessi all'account di gestione.
2. Fai clic su Abilitazione dell'accesso attendibile.
   a. Ora puoi vedere il pannello di controllo dall'account di gestione.
3. Fai clic su  Registra amministratore delegato  e inserisci l'ID dell' AWS account che utilizzerai per visualizzare la dashboard.
   a. Ora puoi visualizzare il pannello di controllo dall'account amministratore delegato o dall'account di gestione.

Per informazioni sulle autorizzazioni necessarie per abilitare il pannello di controllo, consulta Policy IAM minime richieste.

## Visualizzazione del pannello di controllo EKS

1. Accedi all'account amministratore delegato (consigliato) o all'account di gestione.
2. Accedi alla regione us-east-1.
3. Vai al servizio EKS e seleziona pannello di controllo dalla barra laterale sinistra.

> **Note**
>
> Controlla le autorizzazioni IAM necessarie per visualizzare il pannello di controllo EKS.

# Configura il pannello di controllo

Puoi configurare la visualizzazione del pannello di controllo e filtrare le risorse.

## Risorse disponibili

- Cluster: visualizza informazioni aggregate relative allo stato e alla posizione dei cluster EKS.
  - Cluster con problemi di integrità.
  - Cluster su EKS Extended Support.
  - Suddivisione dei cluster sulla base della versione di Kubernetes.
- Gruppi di nodi gestiti: esamina i gruppi di nodi gestiti e le istanze EC2.
  - Gruppi di nodi per tipo di AMI, come Amazon Linux o Bottlerocket.
  - Problemi di integrità dei gruppi di nodi.
  - Distribuzione dei tipi di istanza
- Add-ons: scopri quali Amazon EKS Add-ons hai installato e il loro stato.
  - Numero di installazioni per componente aggiuntivo.
  - Add-ons con problemi di salute.
  - Distribuzione della versione per componente aggiuntivo.

## Visualizzazioni disponibili

- Vista grafico
  - Una vista widget personalizzabile che presenta grafici e visualizzazioni della risorsa selezionata.
  - Le modifiche alla vista grafico, come la rimozione di un widget, sono visibili a tutti gli utenti del pannello di controllo EKS.
- Vista risorsa
  - Una vista a elenco della risorsa selezionata, che supporta i filtri.
- Vista mappa
  - Mostra la distribuzione geografica della risorsa selezionata.

## Filtrare il pannello di controllo EKS

Puoi filtrare il pannello di controllo EKS per:

- AWS Account
- Unità organizzativa, definita da AWS Organizzazioni
- AWS Regione

## Disabilita il pannello di controllo EKS usando il AWS console

1. Conferma quanto segue:
   a. AWS Le organizzazioni sono abilitate e configurate.
   b. Hai effettuato l'accesso all'account di gestione dell'organizzazione.
   c. Stai visualizzando la console AWS di gestione nella regione us-east-1.
2. Passa alla console EKS.
3. Nella barra laterale sinistra, apri Impostazioni del pannello di controllo.
4. Fai clic su Disabilita accesso attendibile.

## Risoluzione dei problemi del pannello di controllo EKS

### Problema di attivazione del pannello di controllo EKS

- È necessario accedere all'account di gestione di un'organizzazione. AWS
  - Se non hai un' AWS organizzazione, creane una. Scopri come Creare e configurare un'organizzazione.
  - Se il tuo AWS account è già membro di un' AWS organizzazione, identifica l'amministratore dell'organizzazione.
- Devi aver effettuato l'accesso all' AWS account con autorizzazioni IAM sufficienti per creare e aggiornare le risorse dell' AWS organizzazione.

### Problema nella visualizzazione del pannello di controllo EKS

- Devi aver effettuato l'accesso a uno dei seguenti account: AWS
  - L'account di gestione dell' AWS organizzazione
  - Un account amministratore delegato, identificato nelle impostazioni del pannello di controllo EKS dell'account di gestione.

- Se hai recentemente abilitato il pannello di controllo EKS, tieni presente che la compilazione iniziale dei dati può richiedere fino a 12 ore.
- Prova a riattivare il pannello di controllo usando la CLI, inclusa la creazione del ruolo collegato al servizio.

## I widget del pannello di controllo si spostano in modo imprevisto

- L'EKS Dashboard salva la visualizzazione del widget configurabile a livello di AWS Account. Se modifichi la visualizzazione del widget, le altre persone che utilizzano lo stesso AWS account vedranno le modifiche.

## Configurare l'integrazione di EKS Dashboard con AWS Organizations

Questa sezione fornisce step-by-step istruzioni per configurare l'integrazione di EKS Dashboard con AWS Organizations. Imparerai come abilitare e disabilitare l'accesso attendibile tra i servizi, nonché come registrare e cancellare gli account di amministratore delegato. Ogni attività di configurazione può essere eseguita utilizzando la AWS console o la AWS CLI.

Abilitazione dell'accesso attendibile

L'accesso attendibile autorizza il pannello di controllo EKS ad accedere in modo sicuro alle informazioni sui cluster di tutti gli account della tua organizzazione.

Utilizzo della console AWS

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.
3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Fai clic su Abilitazione dell'accesso attendibile.

> **Note**
>
> Quando abiliti l'accesso attendibile tramite la console EKS, il sistema crea automaticamente il ruolo collegato al servizio `AWSServiceRoleForAmazonEKSDashboard`. Questa creazione automatica non si verifica se si abilita l'accesso affidabile utilizzando la AWS CLI o la console AWS Organizations.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui i comandi seguenti:

```
aws iam create-service-linked-role --aws-service-name dashboard.eks.amazonaws.com
aws organizations enable-aws-service-access --service-principal eks.amazonaws.com
```

Disabilitare l'accesso attendibile

La disabilitazione dell'accesso attendibile revoca l'autorizzazione del pannello di controllo EKS ad accedere alle informazioni sui cluster negli account dell'organizzazione.

Utilizzo della AWS console

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.
3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Fai clic su Disabilita accesso attendibile.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui il comando seguente:

```
aws organizations disable-aws-service-access --service-principal eks.amazonaws.com
```

Abilitazione di un account di amministratore delegato

Un amministratore delegato è un account membro a cui è stata concessa l'autorizzazione ad accedere al pannello di controllo EKS.

Utilizzo della AWS console

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.

3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Fai clic su Registrazione degli amministratori delegati.
5. Inserisci l'ID account dell' AWS account che desideri scegliere come amministratore delegato.
6. Conferma la registrazione.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui il seguente comando, sostituendo `123456789012` con l'ID del tuo account:

```
aws organizations register-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Disabilitare un account di amministratore delegato

La disabilitazione di un amministratore delegato rimuove l'autorizzazione dell'account ad accedere al pannello di controllo EKS.

Utilizzo della AWS console

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.
3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Trova l'amministratore delegato nell'elenco.
5. Fai clic su Annulla registrazione vicino all'account che vuoi rimuovere come amministratore delegato.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui il seguente comando, sostituendolo `123456789012` con l'ID dell'account dell'amministratore delegato.

```
aws organizations deregister-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

### Sono richieste policy IAM minime

Questa sezione descrive le politiche IAM minime richieste per consentire l'accesso affidabile e delegare un amministratore per l'integrazione di EKS Dashboard con Organizations AWS .

#### Policy per consentire l'accesso attendibile

Per abilitare l'accesso affidabile tra EKS Dashboard e AWS Organizations, sono necessarie le seguenti autorizzazioni:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
dashboard.eks.amazonaws.com/AWSServiceRoleForAmazonEKSDashboard"
        }
    ]
}
```

#### Policy per la delega di un amministratore

Per registrare o annullare la registrazione di un amministratore delegato per il pannello di controllo EKS, servono le seguenti autorizzazioni:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "organizations:RegisterDelegatedAdministrator",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy per la visualizzazione del pannello di controllo EKS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSDashboardReadOnly",
            "Effect": "Allow",
            "Action": [
                "eks:ListDashboardData",
                "eks:ListDashboardResources",
                "eks:DescribeClusterVersions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsReadOnly",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListRoots",
                "organizations:ListAccountsForParent",
                "organizations:ListOrganizationalUnitsForParent"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsDelegatedAdmin",
            "Effect": "Allow",
            "Action": [
                "organizations:ListDelegatedAdministrators"
            ],
```

```
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "organizations:ServicePrincipal": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> Queste policy devono essere collegate al principale IAM (utente o ruolo) nell'account di gestione dell' AWS organizzazione. Gli account dei membri non possono consentire l'accesso attendibile o delegare amministratori.

# Migliora EKS con AWS servizi integrati

Oltre ai servizi descritti in altre sezioni, Amazon EKS collabora con altri AWS servizi per fornire soluzioni aggiuntive. Questo argomento identifica alcuni degli altri servizi che utilizzano Amazon EKS per aggiungere funzionalità o i servizi utilizzati da Amazon EKS per eseguire le attività.

Argomenti


- Effettua il backup dei tuoi cluster EKS con AWS Backup
- Creare risorse Amazon EKS con AWS CloudFormation
- Connect ai repository Git con AWS CodeConnections
- Analizza gli eventi di sicurezza su EKS con Amazon Detective
- Rileva le minacce con Amazon GuardDuty
- Avvia cluster EKS a bassa latenza con AWS Local Zones
- Valutare la resilienza dei cluster EKS con AWS Resilience Hub
- Gestisci i segreti delle applicazioni con AWS Secrets Manager
- Centralizza e analizza i dati di sicurezza EKS con Security Lake
- Abilitare una connettività sicura tra cluster con Amazon VPC Lattice


## Effettua il backup dei tuoi cluster EKS con AWS Backup

AWS Backup supporta i backup dei cluster Amazon EKS, incluso lo stato del cluster Kubernetes e lo storage persistente collegato al cluster EKS tramite una dichiarazione di volume persistente (volumi EBS, file system EFS e bucket S3). Un backup di Amazon EKS creerà un punto di ripristino composito, in cui verrà inserito un punto di ripristino secondario per ogni risorsa sottoposta a backup.

- Come eseguire il backup delle risorse: Guida introduttiva al AWS backup
- Creazione del backup per tipo di risorsa: Amazon EKS Backups
- Come ripristinare i cluster Amazon EKS: ripristino di Amazon EKS Backups

Per iniziare tramite la console EKS, la console di AWS backup o la CLI, assicurati che il tuo ruolo IAM disponga delle seguenti autorizzazioni:

- Queste possono essere trovate nella policy gestita di AWS Backup. AWSBackupServiceRolePolicyForBackup Contiene le autorizzazioni necessarie per il backup del cluster Amazon EKS e dello storage persistente EBS ed EFS
- Se il tuo cluster EKS contiene un bucket S3, dovrai assicurarti che le seguenti politiche e prerequisiti per il tuo bucket S3 siano aggiunti e abilitati come documentato:
- AWSBackupServiceRolePolicyForS3Backup
- Prerequisiti per S3 Backups https://docs.aws.amazon.com/aws-backup/latest/devguide/s3-backups.html#s3-backup-prerequisites

Assicurati che i tuoi cluster EKS abbiano le seguenti impostazioni:

- La modalità di autorizzazione del cluster EKS è impostata su API o AWS API_AND_CONFIG_MAP for Backup per creare voci di accesso per accedere al cluster EKS.

# Creare risorse Amazon EKS con AWS CloudFormation

Amazon EKS è integrato con AWS CloudFormation, un servizio che consente di modellare e configurare le risorse AWS in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. # possibile creare un modello che descriva tutte le risorse AWS desiderate, ad esempio un cluster Amazon EKS. In questo modo, sarà AWS CloudFormation a gestire il provisioning e la configurazione delle risorse.

Quando si utilizza AWS CloudFormation, è possibile riutilizzare il modello per configurare le risorse Amazon EKS in modo coerente e continuo. È sufficiente descrivere le risorse una volta e quindi allestire le stesse risorse più volte in più regioni e account AWS.

## Modelli Amazon EKS e AWS CloudFormation

# necessario conoscere i modelli AWS CloudFormation, prima di poter effettuare il provisioning e la configurazione delle risorse per Amazon EKS e per i servizi correlati. I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse di cui si vuole effettuare il provisioning negli stack AWS CloudFormation. Se non si ha familiarità con JSON o YAML, è possibile usare AWS CloudFormation Designer per iniziare a utilizzare i modelli AWS CloudFormation. Per ulteriori informazioni, consultare la pagina What is AWS CloudFormation Designer? della Guida per l'utente di AWS CloudFormation.

Amazon EKS supporta la creazione di cluster e gruppi di nodi in AWS CloudFormation. Per ulteriori informazioni, inclusi esempi di modelli JSON e YAML per le risorse Amazon EKS, consultare la pagina Amazon EKS resource type reference della Guida per l'utente di AWS CloudFormation.

### Ulteriori informazioni su AWS CloudFormation

Per ulteriori informazioni su AWS CloudFormation, consultare le seguenti risorse:

- AWS CloudFormation
- Guida per l'utente di AWS CloudFormation
- Guida per l'utente dell'interfaccia a riga di comando di AWS CloudFormation

## Connect ai repository Git con AWS CodeConnections

AWS CodeConnectionsfornisce un modo sicuro per connettere i AWS servizi a repository di codice sorgente di terze parti. AWS CodeConnections supporta GitHub GitLab, Bitbucket e altri provider. Per saperne di più e iniziare, consulta Lavorare con le connessioni.

### Utilizzare AWS CodeConnections con Argo CD

Quando si utilizza la funzionalità EKS per Argo CD, è possibile scegliere di AWS CodeConnections abilitare l'autenticazione sicura nei repository Git senza gestire credenziali di lunga durata o token di accesso personali. AWS CodeConnections gestisce il flusso di OAuth autenticazione e gestisce la connessione al provider Git, fornendo un approccio sicuro e gestibile all'accesso ai GitOps repository e ai manifesti delle applicazioni archiviati in provider Git di terze parti.

Prerequisiti

- Creazione di un cluster Amazon EKS con funzionalità Argo CD
- Una connessione creata nel AWS CodeConnections tuo provider Git
- Autorizzazioni IAM configurate per Argo CD per utilizzare la connessione

Per configurare l'accesso al CodeConnections repository di Argo CD

1. Crea una connessione nella console: CodeConnections
   a. Apri la CodeConnections console.

b. Scegli Crea connessione.

c. Seleziona il tuo provider (GitHubo Bitbucket) e segui il flusso di autenticazione. GitLab

d. Nota l'ARN di connessione da utilizzare nella configurazione del CD Argo.

2. Assicurati che il ruolo di funzionalità di Argo CD disponga delle autorizzazioni per utilizzare la connessione con una politica basata sulle risorse:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection",
        "codeconnections:GetConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-
id"
    }
  ]
}
```

3. Configura Argo CD in modo che faccia riferimento alla CodeConnections risorsa quando aggiungi un repository, usando l'endpoint della risorsa come URL del CodeConnections repository. La funzionalità Argo CD utilizza la connessione per autenticarsi al provider Git senza richiedere credenziali di lunga durata.

## Considerazioni sull'utilizzo con Argo CD CodeConnections

Quando si utilizza AWS CodeConnections con il CD EKS Capability for Argo, tenete presente quanto segue:

- La CodeConnections connessione deve trovarsi nella stessa AWS regione del cluster EKS
- Il ruolo di funzionalità di Argo CD deve disporre di autorizzazioni `codeconnections:UseConnection` e `codeconnections:GetConnection`
- CodeConnections gestisce automaticamente il OAuth flusso e il ciclo di vita delle credenziali

Per ulteriori informazioni sulla configurazione dell'accesso al repository con Argo CD, consulta. the section called “Configura i repository”

# Analizza gli eventi di sicurezza su EKS con Amazon Detective

Amazon Detective consente di analizzare, esaminare e identificare rapidamente la causa principale degli esiti di sicurezza o delle attività sospette. Detective raccoglie automaticamente i dati di registro dalle tue AWS risorse. Utilizza quindi il machine learning, l'analisi statistica e la teoria dei grafi per generare visualizzazioni che consentono di condurre indagini sulla sicurezza più rapide ed efficaci. Le aggregazioni di dati, i riepiloghi e il contesto predefiniti di Detective facilitano e velocizzano l'analisi e la determinazione della natura e dell'estensione dei possibili problemi di sicurezza. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon Detective.

Detective organizza Kubernetes e i AWS dati in risultati come:

- Dettagli del cluster Amazon EKS, tra cui l'identità IAM che ha creato il cluster e il ruolo di servizio del cluster. Puoi esaminare l'attività dell'API Kubernetes AWS e di queste identità IAM con Detective.
- Dettagli del container, come l'immagine e il contesto di sicurezza. Puoi esaminare i dettagli relativi ai pod terminati.
- Attività API di Kubernetes, incluse le tendenze generali di attività API e i dettagli su chiamate API specifiche. Ad esempio, puoi mostrare il numero di chiamate API di Kubernetes riuscite e non riuscite emesse in un intervallo di tempo selezionato. Inoltre, la sezione sulle chiamate API appena osservate potrebbe essere utile per identificare attività sospette.

I log di audit di Amazon EKS sono un pacchetto di origini dati facoltativo che può essere aggiunto al grafico del comportamento di Detective. Puoi visualizzare i pacchetti di origine facoltativi disponibili e il rispettivo stato dal tuo account. Per ulteriori informazioni, consulta Log di audit di Amazon EKS per Detective nella Guida per l'utente di Amazon Detective.

## Uso di Amazon Detective con Amazon EKS

Prima di poter esaminare i risultati, Detective deve essere abilitato per almeno 48 ore nella stessa AWS regione in cui si trova il cluster. Per ulteriori informazioni, consulta le sezioni Configurazione di Amazon Detective nella Guida per l'utente di Amazon Detective.

1. Apri la console Detective all'indirizzo https://console.aws.amazon.com/detective/.
2. Seleziona Cerca nel riquadro di navigazione a sinistra.
3. Seleziona Scegli il tipo, quindi seleziona il Cluster EKS.

4. Inserisci il nome o l'ARN del cluster e quindi scegli Cerca.
5. Nei risultati della ricerca, seleziona il nome del cluster del quale desideri visualizzare l'attività. Per ulteriori informazioni su ciò che puoi visualizzare, consulta Overall Kubernetes API activity involving an Amazon EKS cluster nella Guida per l'utente di Amazon Detective.

## Rileva le minacce con Amazon GuardDuty

Amazon GuardDuty è un servizio di rilevamento delle minacce che aiuta a proteggere account, contenitori, carichi di lavoro e dati all'interno del tuo AWS ambiente. Utilizzando modelli di machine learning (ML) e funzionalità di rilevamento di anomalie e minacce, monitora GuardDuty continuamente diverse fonti di log e attività di runtime per identificare e dare priorità ai potenziali rischi per la sicurezza e alle attività dannose nel tuo ambiente.

Tra le altre funzionalità, GuardDuty offre le seguenti due funzionalità che rilevano potenziali minacce ai cluster EKS: EKS Protection e Runtime Monitoring.

> **Note**
>
> Novità: la modalità automatica di Amazon EKS si integra con GuardDuty.

Protezione EKS

Questa funzionalità fornisce una copertura di rilevamento delle minacce per aiutarti a proteggere i cluster Amazon EKS monitorando i log di audit Kubernetes associati. I log di audit Kubernetes acquisiscono azioni sequenziali all'interno del cluster, comprese attività di utenti, applicazioni che utilizzano l'API Kubernetes e il piano di controllo. Ad esempio, è in GuardDuty grado di identificare che le API chiamate a manomettere potenzialmente le risorse in un cluster Kubernetes sono state richiamate da un utente non autenticato.

Quando attivi EKS Protection, GuardDuty potrai accedere ai log di controllo di Amazon EKS solo per il rilevamento continuo delle minacce. Se GuardDuty identifica una potenziale minaccia per il cluster, genera un risultato del log di controllo Kubernetes associato di un tipo specifico. Per ulteriori informazioni sui tipi di risultati disponibili nei log di controllo di Kubernetes, consulta i tipi di ricerca dei log di controllo di Kubernetes nella Amazon User Guide. GuardDuty

Per ulteriori informazioni, consulta EKS Protection nella Amazon GuardDuty User Guide.

Monitoraggio del runtime

Questa funzionalità monitora e analizza gli eventi a livello di sistema operativo, di rete e di file per aiutarti a rilevare potenziali minacce in carichi di AWS lavoro specifici del tuo ambiente.

Quando abiliti il monitoraggio del runtime e installi l' GuardDuty agente nei tuoi cluster Amazon EKS, GuardDuty inizia a monitorare gli eventi di runtime associati a questo cluster. Tieni presente che l' GuardDuty agente e il monitoraggio del runtime non sono disponibili per i nodi ibridi Amazon EKS, quindi il monitoraggio del runtime non è disponibile per gli eventi di runtime che si verificano sui nodi ibridi. Se GuardDuty identifica una potenziale minaccia per il cluster, genera un risultato associato al Runtime Monitoring. Ad esempio, una minaccia può iniziare compromettendo un singolo container che esegue un'applicazione Web vulnerabile. Questa applicazione Web potrebbe disporre delle autorizzazioni di accesso ai container e ai carichi di lavoro sottostanti. In questo scenario, le credenziali configurate in modo errato potrebbero portare a un accesso più ampio all'account e ai dati archiviati al suo interno.

Per configurare Runtime Monitoring, installi l' GuardDuty agente nel tuo cluster come componente aggiuntivo Amazon EKS. Per ulteriori informazioni sul componente aggiuntivo, consulta the section called " AWS componenti aggiuntivi".

Per ulteriori informazioni, consulta Runtime Monitoring nella Amazon GuardDuty User Guide.

# Avvia cluster EKS a bassa latenza con AWS Local Zones

Una zona AWS locale è un'estensione di una AWS regione in prossimità geografica degli utenti. Le zone locali hanno le loro connessioni a Internet e supportano AWS Direct Connect. Le risorse create in una zona locale possono servire gli utenti locali con comunicazioni a latenza molto bassa. Per ulteriori informazioni, consulta AWS Local Zones User Guide e Local Zones nella Guida per l'utente di Amazon EC2.

Amazon EKS supporta alcune risorse nelle zone locali, tra cui i gruppi di nodi gestiti, i nodi Amazon EC2 autogestiti, i volumi Amazon EBS e gli Application Load Balancer (ALB). Si consiglia di prendere in considerazione quanto segue nel momento in cui si utilizzano le zone locali come parte del cluster Amazon EKS.

- Non puoi creare nodi Fargate nelle zone locali con Amazon EKS.
- Il piano di controllo Kubernetes gestito da Amazon EKS funziona sempre nella regione. AWS Il piano di controllo Kubernetes gestito da Amazon EKS non può essere eseguito nella zona

locale. Poiché le Local Zones vengono visualizzate come sottoreti all'interno del VPC, Kubernetes considera le risorse della zona locale come parte di tale sottorete.

- Il cluster Amazon EKS Kubernetes comunica con le istanze Amazon EC2 eseguite nella AWS regione o nella zona locale utilizzando interfacce di rete elastiche gestite da Amazon EKS. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html Per ulteriori informazioni sull'architettura di rete Amazon EKS, consulta Configurazione delle reti.
- A differenza delle sottoreti regionali, Amazon EKS non può inserire interfacce di rete elastiche tra account nelle sottoreti della zona locale. Queste interfacce vengono utilizzate per la comunicazione tra piano di controllo e nodo. Ciò significa che non è necessario specificare sottoreti della zona locale quando si crea il cluster. Tuttavia, è possibile avere nodi di lavoro in più zone locali connessi allo stesso cluster.

# Valutare la resilienza dei cluster EKS con AWS Resilience Hub

AWS Resilience Hub valuta la resilienza di un cluster Amazon EKS analizzandone l'infrastruttura. AWS Resilience Hub utilizza la configurazione relativa al controllo degli accessi basato sul ruolo (RBAC) di Kubernetes per valutare i carichi di lavoro Kubernetes implementati nel cluster. Per ulteriori informazioni, consulta Enabling AWS Resilience Hub access to your Amazon EKS cluster nella Guida per l'utente di AWS Resilience Hub.

# Gestisci i segreti delle applicazioni con AWS Secrets Manager

AWS Secrets Manager ti aiuta a gestire, accedere e ruotare credenziali, chiavi API e altri segreti durante il loro ciclo di vita. Con Secrets Manager, puoi proteggere e gestire i segreti utilizzati per accedere alle risorse in AWS, su servizi di terze parti e in locale. Per ulteriori informazioni, consulta la Guida per l'utente di AWS Secrets Manager.

Quando si utilizza il CD EKS Capability for Argo, Secrets Manager offre un modo sicuro per archiviare e recuperare le credenziali del repository Git senza codificare i dati sensibili nella configurazione e nelle risorse del CD Argo. Questa integrazione è particolarmente utile per gestire i token di accesso ai repository privati e le chiavi SSH utilizzate da Argo CD per sincronizzare le applicazioni dai repository Git.

## Usare AWS Secrets Manager con Argo CD

Quando si utilizza il CD EKS Capability for Argo, è possibile memorizzare le credenziali del repository Git in Secrets Manager e configurare Argo CD per recuperarle. Questo approccio è più sicuro rispetto

all'archiviazione delle credenziali direttamente nella configurazione di Argo CD o all'utilizzo di token di accesso personali di lunga durata.

Prerequisiti

- Un cluster Amazon EKS con funzionalità Argo CD abilitata
- Credenziali del repository Git archiviate in AWS Secrets Manager
- Autorizzazioni IAM configurate per Argo CD per accedere a Secrets Manager

Per configurare Argo CD per utilizzare Secrets Manager per le credenziali del repository

1. Memorizza le tue credenziali Git in Secrets Manager. Ad esempio, per memorizzare un token di accesso GitHub personale:

```
aws secretsmanager create-secret \
  --name argocd/github-token \
  --secret-string '{"username":"git","password":"ghp_xxxxxxxxxxxx"}'
```

2. Assicurati che il ruolo di funzionalità di Argo CD disponga delle autorizzazioni per recuperare il segreto:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:region:account-id:secret:argocd/github-
token*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:region:account-id:key/*",
      "Condition": {
        "StringLike": {
```

```
          "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:region:account-
id:secret:argocd/*",
          "kms:ViaService": "secretsmanager.*.amazonaws.com"
        }
      }
    }
  ]
}
```

> **Note**
>
> L'autorizzazione di decrittografia KMS è richiesta perché Secrets Manager crittografa tutti i segreti utilizzando KMS. AWS La condizione limita la decrittografia ai soli segreti con il prefisso. `argocd/` Se si utilizza la chiave AWS gestita predefinita per Secrets Manager, questa autorizzazione è sufficiente. Per le chiavi KMS gestite dal cliente, aggiorna il `Resource` campo con la chiave ARN specifica.

3. Configura Argo CD per utilizzare le credenziali di Secrets Manager. Per informazioni sulla sincronizzazione dei segreti da Secrets Manager con i segreti di Kubernetes a cui Argo CD può fare riferimento, consulta Secret Management nella documentazione di Argo CD.
4. Crea una configurazione del repository Argo CD che faccia riferimento all'ARN segreto:

```
apiVersion: v1
kind: Secret
metadata:
  name: private-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/org/repo
  secretArn: arn:aws:secretsmanager:region:account-id:secret:argocd/github-token
```

Per ulteriori informazioni sulla configurazione dell'accesso al repository con Argo CD, vedere. the section called “Configura i repository”

# Centralizza e analizza i dati di sicurezza EKS con Security Lake

Amazon Security Lake è un servizio di sicurezza di data lake completamente gestito che consente di centralizzare i dati di sicurezza provenienti da varie origini, incluso Amazon EKS. Integrando Amazon EKS con Security Lake, puoi ottenere informazioni più approfondite sulle attività eseguite sulle tue risorse Kubernetes e migliorare il livello di sicurezza dei tuoi cluster Amazon EKS.

> **Note**
>
> Per ulteriori informazioni sull'uso di Security Lake con Amazon EKS e sulla configurazione delle origini dati, consultare la documentazione di Amazon Security Lake.

## Vantaggi dell'utilizzo di Security Lake con Amazon EKS

Dati di sicurezza centralizzati: Security Lake raccoglie e centralizza automaticamente i dati di sicurezza dai cluster Amazon EKS, insieme ai dati provenienti da altri AWS servizi, provider SaaS, fonti locali e fonti di terze parti. Ciò fornisce una visione completa della posizione di sicurezza dell'intera organizzazione.

Formato dati standardizzato: Security Lake converte i dati raccolti nel formato Open Cybersecurity Schema Framework (OCSF), che è uno schema open source standard. Questa normalizzazione consente un'analisi e un'integrazione più semplici con altri strumenti e servizi di sicurezza.

Rilevamento delle minacce migliorato: analizzando i dati di sicurezza centralizzati, inclusi i log del piano di controllo di Amazon EKS, è possibile rilevare attività potenzialmente sospette all'interno dei cluster Amazon EKS in modo più efficace. Questo aiuta a identificare e rispondere tempestivamente agli incidenti di sicurezza.

Gestione semplificata dei dati: Security Lake gestisce il ciclo di vita dei dati di sicurezza con impostazioni di conservazione e replica personalizzabili. Ciò semplifica le attività di gestione dei dati e garantisce la conservazione dei dati necessari per scopi di conformità e controllo.

## Abilitare Security Lake per Amazon EKS

1. Abilita la registrazione del piano di controllo di Amazon EKS per i tuoi cluster EKS. Per istruzioni dettagliate, consulta Enabling and disabling control plane logs.

2. Add Amazon EKS Audit Logs as a source in Security Lake. Security Lake inizierà quindi a raccogliere informazioni approfondite sulle attività eseguite sulle risorse Kubernetes in esecuzione nei cluster EKS.
3. Configure retention and replication settings per i dati di sicurezza in Security Lake in base alle tue esigenze.
4. Utilizza i dati OCSF normalizzati archiviati in Security Lake per la risposta agli incidenti, l'analisi della sicurezza e l'integrazione con altri servizi o strumenti di terze parti. AWS Ad esempio, puoi generare informazioni sulla sicurezza dai dati di Amazon Security Lake utilizzando Amazon Ingestion. OpenSearch

## Analisi dei log EKS in Security Lake

Security Lake normalizza gli eventi del log EKS nel formato OCSF, semplificando l'analisi e la correlazione dei dati con altri eventi di sicurezza. Puoi utilizzare vari strumenti e servizi, come Amazon Athena, Amazon QuickSight o strumenti di analisi di sicurezza di terze parti, per interrogare e visualizzare i dati normalizzati.

Per ulteriori informazioni sulla mappatura OCSF per gli eventi di log EKS, consulta il riferimento di mappatura nel repository OCSF. GitHub

# Abilitare una connettività sicura tra cluster con Amazon VPC Lattice

Amazon VPC Lattice è un servizio di rete di applicazioni completamente gestito integrato direttamente nell'infrastruttura di rete AWS che puoi utilizzare per connettere, proteggere e monitorare i tuoi servizi su più account e cloud privati virtuali (VPC). Con Amazon EKS, puoi sfruttare Amazon VPC Lattice tramite l'uso del Controller API Gateway AWS, un'implementazione dell'API Gateway Kubernetes. Utilizzando Amazon VPC Lattice, è possibile configurare la connettività tra cluster con semantica Kubernetes standard in modo semplice e coerente. Per iniziare a usare Amazon VPC Lattice con Amazon EKS, consulta la Guida per l'utente del controller API Gateway AWS.

# Connettere un cluster Kubernetes a una Console di gestione Amazon EKS con Amazon EKS Connector

Puoi utilizzare Amazon EKS Connector per registrare e connettere qualsiasi cluster Kubernetes conforme e visualizzarlo nella console Amazon EKS. AWS Una volta connesso, è possibile visualizzare lo stato, la configurazione e i carichi di lavoro del cluster nella console Amazon EKS. È possibile utilizzare questa funzionalità per visualizzare i cluster connessi nella console Amazon EKS, ma non è possibile gestirli. Amazon EKS Connector richiede un agente su cui è presente un progetto open source. GitHub Per contenuti tecnici aggiuntivi, tra cui domande frequenti e risoluzione dei problemi, consultathe section called "Risoluzione dei problemi del connettore EKS".

Amazon EKS Connector connette i seguenti tipi di cluster Kubernetes ad Amazon EKS.

- On-premises Cluster Kubernetes
- Self-managed cluster in esecuzione su Amazon EC2
- Cluster gestiti da altri provider cloud

## Considerazioni su Amazon EKS Connector

Prima di utilizzare Amazon EKS Connector, attenersi a quanto segue:

- Prima di connettere il cluster ad Amazon EKS, è necessario disporre dei privilegi amministrativi per il cluster Kubernetes.
- Il cluster Kubernetes deve avere nodi (worker) Linux a 64 bit (x86) presenti prima della connessione. I nodi worker ARM non sono supportati.
- È necessario disporre di nodi (worker) nel cluster Kubernetes che abbiano accesso in uscita agli endpoint di Systems Manager `ssm.` e `ssmmessages.`. Per ulteriori informazioni, consulta Systems Manager endpoints nella AWS Guida generale.
- Per impostazione predefinita, è possibile connettere fino a 10 cluster per regione. È possibile richiedere un aumento tramite la console Service Quotas. Per ulteriori informazioni, consultare Richiesta di un aumento di quota.
- Solo le API `RegisterCluster`, `ListClusters`, `DescribeCluster` e `DeregisterCluster` di Amazon EKS sono supportate per i cluster Kubernetes esterni.
- È necessario disporre delle autorizzazioni seguenti per registrare un cluster:

- Chiavi: RegisterCluster
- sms: CreateActivation
- sms: DeleteActivation
- sono: PassRole

- È necessario disporre delle autorizzazioni seguenti per annullare la registrazione di un cluster:
  - Ex: DeregisterCluster
  - sms: DeleteActivation
  - sms: DeregisterManagedInstance

## Ruoli IAM richiesti per Amazon EKS Connector

L'utilizzo di Amazon EKS Connector richiede i seguenti due ruoli IAM:

- Il ruolo Amazon EKS Connectore collegato al servizio viene creato al momento della prima registrazione del cluster.
- È indispensabile creare il ruolo IAM dell'agente Amazon EKS Connector. Per informazioni dettagliate, vedi the section called "Ruolo IAM di Connector".

Per abilitare l'autorizzazione alla visualizzazione del cluster e del carico di lavoro per i principali IAM, è necessario applicare il `eks-connector` e i ruoli del cluster Amazon EKS Connector nel cluster. Seguire la procedura descritta alla pagina Grant access to view Kubernetes cluster resources on an Amazon EKS console.

## Connettere un cluster Kubernetes esterno alla Console di gestione Amazon EKS

È possibile collegare un cluster Kubernetes esterno ad Amazon EKS utilizzando più metodi, come illustrato nel seguente processo. Questo processo prevede due passaggi: registrare il cluster con Amazon EKS e installare l'agente `eks-connector` nel cluster.

> **Important**
>
> È necessario completare il secondo passaggio entro 3 giorni dal completamento del primo passaggio, ossia prima che la registrazione scada.

## Considerazioni

È possibile utilizzare i manifesti YAML durante l'installazione dell'agente. In alternativa, puoi usare Helm se registri il cluster con Console di gestione AWS o AWS Command Line Interface. Tuttavia, non è possibile utilizzare Helm per installare l'agente se si registra il cluster con `eksctl`.

## Prerequisiti

- Assicurati di avere creato il ruolo agente del connettore Amazon EKS. Seguire i passaggi descritti alla pagina Creating the Amazon EKS connector agent role.
- È necessario disporre delle autorizzazioni seguenti per registrare un cluster:
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

## Fase 1: Registrazione del cluster

Per registrare un cluster su Amazon EKS Connector, puoi utilizzare uno di questi strumenti:

- the section called "AWS CLI"
- the section called "Console di gestione AWS"
- the section called "`eksctl`"

### AWS CLI

1. AWS La CLI deve essere installata. Per installarlo o aggiornarlo, vedi Installazione della AWS CLI.
2. Per la configurazione del plug-in, specificare il ruolo IAM dell'agente Amazon EKS Connector. Per ulteriori informazioni, consulta the section called "Ruoli IAM richiesti per Amazon EKS Connector".

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws: iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

Di seguito viene riportato un output di esempio.

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws: eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws: iam::111122223333:role/AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

I valori `aws-region`, `activationId`, e `activationCode` verranno utilizzati nel prossimo passaggio.

## Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Add cluster (Aggiungi cluster) e seleziona Register (Registra) per visualizzare la pagina di configurazione.
3. Nella pagina Configura cluster compilare i campi riportati di seguito:
   - Nome: un nome univoco per il cluster.
   - Provider: scegli per visualizzare l'elenco a discesa dei provider del cluster Kubernetes. Se non si conosce il provider specifico, selezionare Altro.
   - Ruolo plug-in EKS: selezionare il ruolo da utilizzare per la connessione del cluster.
4. Selezionare Registra cluster.
5. Viene visualizzata la pagina Panoramica del cluster. Se desideri utilizzare il grafico Helm, copia il comando `helm install` e prosegui con il passaggio successivo. Se desideri utilizzare il manifesto YAML, scegli Scarica file YAML per scaricare il file manifesto sull'unità locale.

> **Important**
>
> Questa è l'unica occasione per copiare il comando `helm install` o eseguire il download di questo file. Non uscire da questa pagina, poiché il link non sarà accessibile e sarà necessario annullare la registrazione del cluster e riavviare i passaggi dall'inizio.

Il comando o il file manifesto può essere utilizzato una sola volta per il cluster registrato. Se si eliminano risorse dal cluster Kubernetes, è necessario registrare nuovamente il cluster e ottenere un nuovo file manifesto.

Continua con la fase successiva per applicare il file manifesto al cluster Kubernetes.

### `eksctl`

1. Deve essere installato `eksctl` versione `0.68` o successiva. Per installarlo o aggiornarlo, consulta la pagina the section called "Crea cluster (eksctl)".
2. Registra il cluster fornendo un nome, un provider e una regione.

```
eksctl register cluster --name my-cluster --provider my-provider --region region-code
```

Output di esempio:

```
2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-connector-
console-dashboard-full-access-group.yaml" give full EKS Console access to IAM
 identity "<aws-arn>", edit if required; read https://eksctl.io/usage/eks-connector
 for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry to connect the cluster
```

In questo modo vengono creati file sul computer locale. Questi file devono essere applicati al cluster esterno entro 3 giorni; in caso contrario, la registrazione scade.

3. In un terminale che può accedere al cluster, applica i file generati:

```
kubectl apply -f eks-connector.yaml,eks-connector-clusterrole.yaml,eks-connector-
console-dashboard-full-access-group.yaml
```

## Fase 2: Installazione dell'agente `eks-connector`

Per installare l'agente `eks-connector`, utilizzare uno dei seguenti strumenti:

- the section called "Helm"
- the section called "YAML"

### Helm

> **Note**
>
> Se il cluster è stato registrato con `eksctl`, utilizzare il metodo con manifesto YAML anziché il metodo con grafico Helm.

1. Se hai utilizzato la AWS CLI nel passaggio precedente, sostituisci il comando `ACTIVATION_CODE` and `ACTIVATION_ID` nel comando seguente rispettivamente con i valori `activationId` `eactivationCode`. Sostituisci la AWS regione `aws-region` con la regione utilizzata nel passaggio precedente. Quindi esegui il comando per installare l'`eks-connector`agente sul cluster registrato:

```
$ helm install eks-connector \
  --namespace eks-connector \
  oci://public.ecr.aws/eks-connector/eks-connector-chart \
  --set eks.activationCode=ACTIVATION_CODE \
  --set eks.activationId=ACTIVATION_ID \
  --set eks.agentRegion=aws-region
```

Se hai usato il comando Console di gestione AWS nel passaggio precedente, usa il comando che hai copiato dal passaggio precedente che contiene questi valori.

2. Verifica l'integrità dell'implementazione del `eks-connector` installato e attendi che lo stato del cluster registrato in Amazon EKS diventi `ACTIVE`.

## YAML

Completare la connessione applicando il file manifesto Amazon EKS Connector al proprio cluster Kubernetes. A tale scopo, è necessario utilizzare i metodi descritti in precedenza. Se il manifesto non viene applicato entro tre giorni, la registrazione di Amazon EKS Connector scade. Se la connessione al cluster scade, la registrazione del cluster deve essere annullata prima di poterlo connettere nuovamente.

1. Eseguire il download del file YAML di Amazon EKS Connector.

```
curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/eks-connector/latest/eks-connector.yaml
```

2. Modifica il file YAML del connettore Amazon EKS per sostituire tutti i riferimenti di `%AWS_REGION%`, `%EKS_ACTIVATION_ID%` e `%EKS_ACTIVATION_CODE%` con i valori `aws-region`, `activationId`, e `activationCode` dell'output del passaggio precedente.

   Il comando di esempio seguente può sostituire questi valori.

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE | base64)~g" eks-connector.yaml
```

   > **Important**
   >
   > Assicurarsi che il codice di attivazione sia in formato base64.

3. In un terminale dotato di accesso al cluster, è possibile applicare il file manifesto aggiornato eseguendo il seguente comando:

```
kubectl apply -f eks-connector.yaml
```

4. Dopo aver applicato al cluster Kubernetes il manifesto e i file YAML dell'associazione dei ruoli di Amazon EKS Connector, verificare che il cluster sia connesso.

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

L'output deve includere `status=ACTIVE`.

5. (Facoltativo) Assegna tag al cluster. Per ulteriori informazioni, consulta the section called "Tagging delle risorse ".

## Fasi successive

In caso di problemi con questi passaggi, consulta la pagina the section called "Risoluzione dei problemi del connettore EKS".

Per concedere ai principali IAM aggiuntivi l'accesso alla console Amazon EKS al fine di visualizzare le risorse Kubernetes in un cluster connesso, consultare the section called "Garantire l'accesso ai cluster".

# Concedere l'accesso per visualizzare le risorse del cluster Kubernetes in una console Amazon EKS

Concedere ai principali IAM l'accesso alla console Amazon EKS per visualizzare informazioni sulle risorse Kubernetes in esecuzione sul cluster connesso.

## Prerequisiti

Il principale IAM utilizzato per accedere alla Console di gestione AWS deve soddisfare i seguenti requisiti:

- Deve disporre dell'autorizzazione IAM `eks:AccessKubernetesApi`.
- L'account di servizio Amazon EKS Connector dovrebbe essere in grado di rappresentare il principale IAM nel cluster. Ciò consente ad Amazon EKS Connector di mappare il principale IAM a un utente Kubernetes.

Per creare e applicare il ruolo del cluster Amazon EKS Connector

1. Eseguire il download del modello di ruolo del cluster `eks-connector`.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-
connector-console-roles/eks-connector-clusterrole.yaml
```

2. Modificare il file YAML del modello di ruolo del cluster. Sostituire i riferimenti di `%IAM_ARN%` con il nome della risorsa Amazon (ARN) del principale IAM.
3. Applicare il ruolo del cluster Amazon EKS Connector YAML al cluster Kubernetes.

```
kubectl apply -f eks-connector-clusterrole.yaml
```

Affinché un principale IAM possa visualizzare le risorse Kubernetes nella console Amazon EKS, deve essere associato a un `role` o `clusterrole` Kubernetes con le autorizzazioni necessarie per leggere le risorse. Per ulteriori informazioni, consulta Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.

Per configurare un principale IAM per accedere al cluster connesso

1. È possibile eseguire il download di questi file manifesto di esempio per creare, rispettivamente, un `clusterrole` e un'`clusterrolebinding` oppure un `role` e un'`rolebinding`:

   Visualizzare le risorse Kubernetes in tutti i namespace

   - Il ruolo del cluster `eks-connector-console-dashboard-full-access-clusterrole` consente di accedere a tutti gli spazi dei nomi e alle risorse visualizzabili nella console. É possibile modificare il nome di `role`, `clusterrole` e il loro legame corrispondente prima di applicarlo al cluster. Utilizzare il seguente comando per eseguire il download di un file di esempio.

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/
     eks-connector-console-roles/eks-connector-console-dashboard-full-access-
     group.yaml
     ```

   Visualizzare le risorse Kubernetes in un namespace specifico

   - Lo spazio dei nomi in questo file è `default`, quindi se si desidera specificare uno spazio dei nomi diverso, modificare il file prima di applicarlo al cluster. Utilizzare il seguente comando per eseguire il download di un file di esempio.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/
eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-
group.yaml
```

2. Modificare il file YAML con accesso completo o con accesso limitato per sostituire i riferimenti di `%IAM_ARN%` con il nome della risorsa Amazon (ARN) del principale IAM.
3. Applicare i file YAML ad accesso completo o limitato al cluster Kubernetes. Sostituire il valore del file YAML con il proprio valore.

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

Per visualizzare le risorse Kubernetes nel cluster connesso, consulta the section called "Accesso alle risorse di cluster". I dati relativi ad alcuni tipi di risorse nella scheda Risorse non sono disponibili per i cluster collegati.

# Annullare la registrazione di un cluster Kubernetes dalla console Amazon EKS

Al termine dell'utilizzo di un cluster connesso, è possibile annullarne la registrazione. Una volta annullata la registrazione, il cluster non sarà più visibile nella console Amazon EKS.

È necessario disporre delle autorizzazioni seguenti per chiamare l'API di deregisterCluster:

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`
- `ssm:DeregisterManagedInstance`

Questo processo prevede due passaggi: annullare la registrazione del cluster con Amazon EKS e disinstallare l'agente eks-connector nel cluster.

## Annullare la registrazione del cluster Kubernetes

Per annullare la registrazione di un cluster da Amazon EKS Connector, puoi utilizzare uno di questi strumenti:

- the section called "AWS CLI"

- the section called "Console di gestione AWS"
- the section called "`eksctl`"

### AWS CLI

1. AWS La CLI deve essere installata. Per installarlo o aggiornarlo, vedi Installazione della AWS CLI.
2. Assicurati di avere creato il ruolo agente del connettore Amazon EKS.
3. Annullare la registrazione del cluster connesso.

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

### Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Cluster.
3. Nella pagina Cluster, selezionare il cluster connesso e selezionare Annulla registrazione.
4. Confermare che si desidera annullare la registrazione del cluster.

### `eksctl`

1. Installare la versione `0.68` o successiva di `eksctl`. Per installarlo o aggiornarlo, consulta la pagina the section called "Crea cluster (eksctl)".
2. Assicurati di avere creato il ruolo agente del connettore Amazon EKS.
3. Annullare la registrazione del cluster connesso:

```
eksctl deregister cluster --name my-cluster
```

## Eliminare le risorse del cluster Kubernetes

Per disinstallare l'agente `eks-connector`, utilizzare uno dei seguenti strumenti:

- the section called "Helm"

- the section called "YAML"

### Helm

Per disinstallare l'agente, esegui il comando seguente.

```
helm -n eks-connector uninstall eks-connector
```

### YAML

1. Eliminare il file YAML di Amazon EKS Connector dal cluster Kubernetes.

   ```
   kubectl delete -f eks-connector.yaml
   ```

2. Se sono stati creati `clusterrole` o `clusterrolebindings` per fornire accesso al cluster a principali IAM aggiuntivi, assicurarsi di eliminarli dal cluster Kubernetes.

## Risoluzione dei problemi del connettore Amazon EKS

In questo argomento sono descritti alcuni degli errori più comuni che si possono verificare durante l'utilizzo del connettore Amazon EKS e sono riportate le istruzioni per risolverli in maniera temporanea o definitiva.

### Risoluzione dei problemi di base

In questa sezione sono descritti i passaggi per la diagnosi dei problemi del connettore Amazon EKS.

#### Verifica dello stato di Amazon EKS Connector

Per controllare lo stato del connettore Amazon EKS, inserisci:

```
kubectl get pods -n eks-connector
```

#### Ispezione dei log del connettore Amazon EKS

Il pod di Amazon EKS Connector è composto da tre container. Per recuperare i registri completi per tutti questi container in modo da poterli ispezionare, eseguire i seguenti comandi:

- `connector-init`

```
kubectl logs eks-connector-0 --container connector-init -n eks-connector
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- `connector-proxy`

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- `connector-agent`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

## Ottenimento del nome effettivo del cluster

I cluster Amazon EKS sono identificati in modo univoco `clusterName` all'interno di un singolo AWS account e AWS regione. Se si hanno più cluster connessi in Amazon EKS, è possibile verificare quale cluster Amazon EKS è registrato con il cluster Kubernetes corrente. Per fare ciò, inserisci quanto segue per individuare il `clusterName` del cluster corrente.

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

## Comandi vari

I seguenti comandi sono utili per recuperare le informazioni necessarie per risolvere i problemi.

- Usa il seguente comando per raccogliere le immagini utilizzate dai Pods in Amazon EKS Connector.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
 | tr -s '[[:space:]]' '\n'
```

- Usa il comando seguente per determinare i nomi dei nodi su cui è in esecuzione il connettore Amazon EKS.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
 '[[:space:]]' '\n'
```

- Per ottenere le versioni client e server Kubernetes esegui il comando riportato.

```
kubectl version
```

- Usa il comando seguente per ottenere informazioni sui tuoi nodi.

```
kubectl get nodes -o wide --show-labels
```

## Problema di Helm: 403 Forbidden

Se hai riscontrato il seguente errore durante l'esecuzione dei comandi di installazione di helm:

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

Puoi eseguire la riga seguente per risolvere il problema:

```
docker logout public.ecr.aws
```

## Errore della console: il cluster è bloccato nello stato Pending (In sospeso)

Se il cluster rimane bloccato nella `Pending` console Amazon EKS dopo averlo registrato, potrebbe essere perché Amazon EKS Connector non è AWS ancora riuscito a connettere il cluster. Per un cluster registrato, lo stato `Pending` implica che la connessione non è stata stabilita correttamente. Per risolvere il problema, assicurati di aver applicato il manifesto al cluster Kubernetes di destinazione. Se l'hai applicato al cluster ma il cluster rimane ancora nello stato `Pending`, lo statefulset `eks-connector` potrebbe non essere integro. Per risolvere questo problema, consulta the section called "I Connector Pod di Amazon EKS si bloccano in modo anomalo" questo argomento.

## Errore della console: User system:serviceaccount:eks-connector:eks-connector non può impersonare la risorsa utenti nel gruppo API nell'ambito del cluster

Il connettore Amazon EKS utilizza la rappresentazione dell'utente di Kubernetes per agire per conto dei principali IAM da Console di gestione AWS. A ogni principale che accede all'API Kubernetes dall'account del AWS `eks-connector` servizio deve essere concessa l'autorizzazione a impersonare l'utente Kubernetes corrispondente con un ARN IAM come nome utente Kubernetes. Negli esempi seguenti, l'ARN IAM viene mappato a un utente Kubernetes.

- *111122223333*L' AWS utente IAM dell'account è mappato a un utente Kubernetes. *john* Le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti.

  ```
  arn:aws: iam::111122223333:user/john
  ```

- Il ruolo IAM dell' AWS account *111122223333* è mappato *admin* a un utente Kubernetes:

  ```
  arn:aws: iam::111122223333:role/admin
  ```

  Il risultato è un ARN del ruolo IAM, anziché l'ARN della sessione STS. AWS

Per le istruzioni per configurare `ClusterRole` e `ClusterRoleBinding` per concedere il privilegio dell'account di servizio `eks-connector` per impersonare l'utente mappato, consulta the section called "Garantire l'accesso ai cluster". Assicurati che nel modello, `%IAM_ARN%` sia sostituito con l'ARN IAM del Console di gestione AWS principale IAM.

## Errore della console: […] non consentito: L'utente […] non può elencare la risorsa […] nel gruppo API nell'ambito del cluster

Consideriamo il seguente problema. Amazon EKS Connector ha impersonato con successo il principale Console di gestione AWS IAM richiedente nel cluster Kubernetes di destinazione. Tuttavia, il principale impersonato non dispone dell'autorizzazione RBAC per le operazioni API di Kubernetes.

Per risolvere questo problema, esistono due metodi per concedere le autorizzazioni ad altri utenti. Se in precedenza hai installato eks-connector tramite un grafico Helm, puoi concedere facilmente l'accesso agli utenti eseguendo il seguente comando. Sostituisci i valori `userARN1` e `userARN2` con un elenco delle ARN dei ruoli IAM a cui consentire l'accesso per visualizzare le risorse Kubernetes:

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

In alternativa, in qualità di amministratore del cluster, concedi il livello appropriato di privilegi RBAC ai singoli utenti Kubernetes. Per maggiori informazioni ed esempi, consulta the section called "Garantire l'accesso ai cluster".

## Errore della console: Amazon EKS non è in grado di comunicare con il server API cluster Kubernetes. Per una connessione corretta, il cluster deve trovarsi nello stato ACTIVE (ATTIVO). Riprova tra qualche minuto.

Se il servizio Amazon EKS non è in grado di comunicare con Amazon EKS Connector nel cluster di destinazione, potrebbe essere dovuto a uno dei seguenti motivi:

- Il connettore Amazon EKS nel cluster di destinazione non è integro.
- Connettività scadente o connessione interrotta tra il cluster di destinazione e la AWS regione.

Per risolvere questo problema, controlla Amazon EKS Connector logs (Log del connettore Amazon EKS). Se non visualizzi un errore per il connettore Amazon EKS, ritenta la connessione dopo qualche minuto. Se riscontri regolarmente una latenza elevata o una connettività intermittente per il cluster di destinazione, valuta la possibilità di registrare nuovamente il cluster in una AWS regione più vicina a te.

## I Connector Pod di Amazon EKS si bloccano in modo anomalo

Sono molte le ragioni che possono far sì che un Amazon EKS Connector Pod entri in questo stato. `CrashLoopBackOff` Questo problema riguarda probabilmente il container `connector-init`. Verifica lo stato di Amazon EKS Connector Pod.

```
kubectl get pods -n eks-connector
```

Di seguito viene riportato un output di esempio.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

Se l'output è simile a quello precedente, consulta la pagina the section called "Ispezione dei log del connettore Amazon EKS" per risolvere il problema.

## Avvio di eks-connector non riuscito: InvalidActivation

Quando avvii il connettore Amazon EKS per la prima volta, il connettore registra un `activationId` e un `activationCode` con Amazon Web Services. La registrazione potrebbe non riuscire, causando un arresto anomalo del container `connector-init` con un errore simile al seguente.

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

Per risolvere questo problema, prendere in considerazione le seguenti cause e le correzioni consigliate:

- La registrazione potrebbe non essere riuscita perché nel file manifesto mancano `activationId` e `activationCode`. In questo caso, accertati che siano i valori corretti restituiti dall'operazione API `RegisterCluster` e che `activationCode` sia presente nel file manifesto. `activationCode` è aggiunto ai segreti Kubernetes, per cui deve essere codificato in `base64`. Per ulteriori informazioni, consulta the section called "Fase 1: Registrazione del cluster".
- La registrazione potrebbe non essere riuscita perché l'attivazione è scaduta. Ciò accade perché, per motivi di sicurezza, devi attivare il connettore Amazon EKS entro tre giorni dalla registrazione del cluster. Per risolvere il problema, assicurati di aver applicato il manifesto del connettore Amazon EKS al cluster Kubernetes di destinazione prima della data e dell'ora di scadenza. Per verificare la data di scadenza dell'attivazione, richiama l'operazione API `DescribeCluster`.

  ```
  aws eks describe-cluster --name my-cluster
  ```

  Nella risposta di esempio seguente, la data e l'ora di scadenza vengono registrate come `2021-11-12T22:28:51.101000-08:00`.

  ```
  {
      "cluster": {
          "name": "my-cluster",
          "arn": "arn:aws: eks:region:111122223333:cluster/my-cluster",
          "createdAt": "2021-11-09T22:28:51.449000-08:00",
          "status": "FAILED",
          "tags": {
          },
  ```

```
        "connectorConfig": {
            "activationId": "00000000-0000-0000-0000-000000000000",
            "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
            "provider": "OTHER",
            "roleArn": "arn:aws: iam::111122223333:role/my-connector-role"
        }
    }
}
```

Se `activationExpiry` è passato, annulla la registrazione del cluster e registralo di nuovo. In questo modo si genera una nuova attivazione.

## Nel nodo del cluster manca la connettività in uscita

Per funzionare correttamente, Amazon EKS Connector richiede la connettività in uscita verso diversi endpoint. AWS Non è possibile connettere un cluster privato senza connettività in uscita a una regione di destinazione. AWS Per risolvere il problema, è necessario aggiungere la connettività in uscita necessaria. Per informazioni sui requisiti del connettore, consultare the section called "Considerazioni su Amazon EKS Connector".

## I Connector Pod di Amazon EKS sono in stato **`ImagePullBackOff`**

Se esegui il comando `get pods` e i pod sono nello stato `ImagePullBackOff`, non possono funzionare correttamente. Se i pod del connettore Amazon EKS sono nello stato `ImagePullBackOff`, non possono funzionare correttamente. Controlla lo stato dei pod del connettore Amazon EKS.

```
kubectl get pods -n eks-connector
```

Di seguito viene riportato un output di esempio.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

Il file manifesto predefinito del connettore Amazon EKS fa riferimento alle immagini di Amazon ECR Public Gallery. È possibile che il cluster Kubernetes di destinazione non possa estrarre immagini da Amazon ECR Public Gallery. Risolvi il problema di estrazione delle immagini di Amazon ECR Public Gallery o valuta il mirroring delle immagini nel registro del container privato di tua scelta.

# Domande frequenti su Amazon EKS Connector

D: Come funziona la tecnologia alla base del connettore Amazon EKS?

R: Amazon EKS Connector si basa sull'agente Systems Manager (Systems Manager). AWS Il connettore Amazon EKS è eseguito come `StatefulSet` nel tuo cluster . Stabilisce una connessione e applica un proxy alla comunicazione tra il server API del cluster e Amazon Web Services. Lo fa per visualizzare i dati del cluster nella console Amazon EKS fino alla disconnessione dal AWS cluster. L'agente Systems Manager è un progetto open source. Per ulteriori informazioni su questo progetto, consulta la pagina del progetto su GitHub .

D: Dispongo di un cluster Kubernetes on-premises a cui desidero connettermi. Devo aprire le porte firewall per connetterlo?

R: No, non è necessario aprire alcuna porta firewall. Il cluster Kubernetes richiede solo una connessione in uscita alle regioni. AWS AWS i servizi non accedono mai alle risorse della tua rete locale. Amazon EKS Connector viene eseguito sul cluster e avvia la connessione a. AWS Una volta completata la registrazione del cluster, AWS invia comandi ad Amazon EKS Connector solo dopo aver avviato un'azione dalla console Amazon EKS che richiede informazioni dal server API Kubernetes sul cluster.

D: Quali dati vengono inviati dal mio cluster a AWS tramite Amazon EKS Connector?

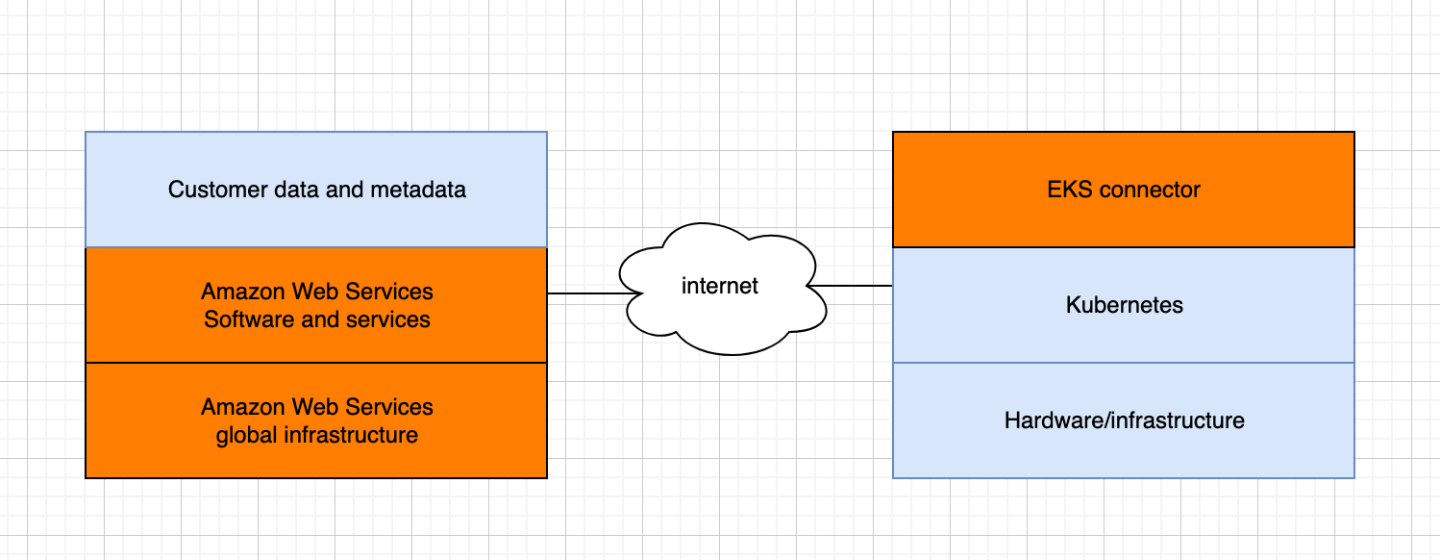
R: Il connettore Amazon EKS invia le informazioni tecniche necessarie per la registrazione del cluster su AWS. Invia inoltre metadati di cluster e carichi di lavoro per le funzionalità della console Amazon EKS richieste dai clienti. Il connettore Amazon EKS raccoglie o invia questi dati solo se avvii un'azione dalla console Amazon EKS o dall'API Amazon EKS che richiede l'invio dei dati a AWS. A parte il numero di versione di Kubernetes, per impostazione AWS predefinita non memorizza alcun dato. Memorizza i dati solo se lo autorizzi.

D: Posso connettere un cluster all'esterno di un AWS Regione?

R: Sì, puoi connettere un cluster da qualsiasi posizione ad Amazon EKS. Inoltre, il tuo servizio Amazon EKS può essere localizzato in qualsiasi AWS regione commerciale pubblica. Funziona con una connessione di rete valida dal cluster alla AWS regione di destinazione. Ti consigliamo di scegliere una AWS regione più vicina alla posizione del cluster per l'ottimizzazione delle prestazioni dell'interfaccia utente. Ad esempio, se hai un cluster in esecuzione a Tokyo, connettilo alla AWS regione di Tokyo (ovvero la `ap-northeast-1` AWS regione) per una bassa latenza. Puoi connettere un cluster da qualsiasi posizione ad Amazon EKS in qualsiasi AWS regione commerciale pubblica, ad eccezione della Cina o GovCloud AWS delle regioni.

# Comprensione della sicurezza in Amazon EKS Connector

Amazon EKS Connector è un componente open source che viene eseguito sul cluster Kubernetes. Questo cluster può trovarsi al di fuori dell'ambiente AWS, sollevando alcune considerazioni in materia di responsabilità di sicurezza. Il diagramma seguente illustra tale configurazione. Il colore arancione indica le responsabilità di AWS e il colore blu le responsabilità del cliente:




Questo argomento illustra le differenze nel modello di responsabilità quando il cluster connesso è esterno ad AWS.

## Responsabilità di AWS

- Gestire, creare e implementare Amazon EKS Connector, un componente open source che viene eseguito sul cluster Kubernetes di un cliente e comunica con AWS.
- Mantenere la sicurezza delle comunicazioni a livello di trasporto e di applicazione tra il cluster Kubernetes connesso e i servizi AWS.

## Responsabilità del cliente

- Garantire la sicurezza specifica del cluster Kubernetes, in particolare seguendo le linee guida riportate di seguito:
  - I segreti di Kubernetes devono essere adeguatamente crittografati e protetti.
  - L'accesso allo spazio dei nomi `eks-connector` deve essere bloccato.

- Configurazione delle autorizzazioni relative al controllo degli accessi basato sul ruolo (RBAC) per gestire l'accesso del principale IAM da AWS. Per istruzioni, consultare the section called "Garantire l'accesso ai cluster".
- Installazione e aggiornamento di Amazon EKS Connector.
- Manutenzione dell'hardware, del software e dell'infrastruttura che supportano il cluster Kubernetes collegato.
- Protezione degli account AWS (ad esempio, tramite la salvaguardia delle credenziali dell'utente root).

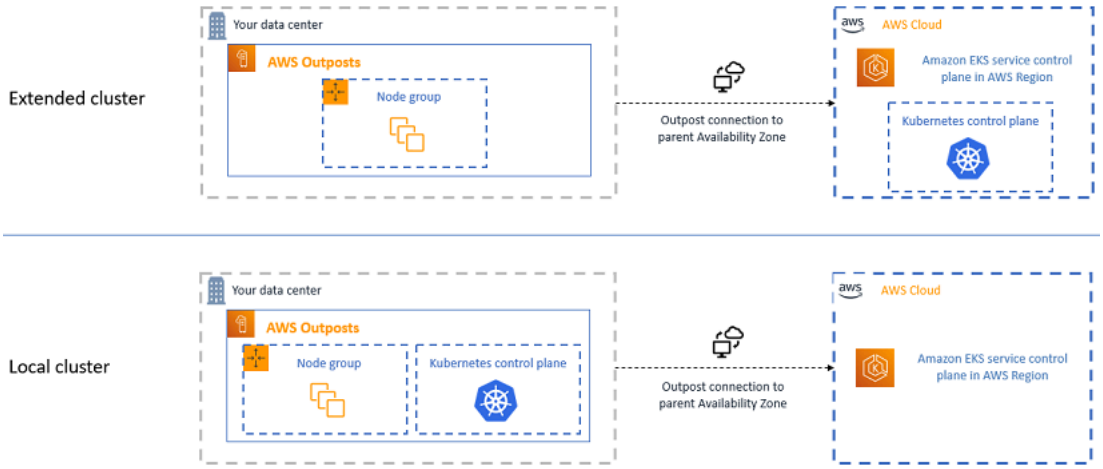
# Implementa Amazon EKS in locale con AWS Outposts

Puoi usare Amazon EKS per eseguire applicazioni Kubernetes locali su Outposts. AWS Puoi implementare Amazon EKS su Outposts nei seguenti modi:

- Cluster estesi: esegui il piano di controllo Kubernetes in una regione e in nodi sul tuo Outpost. AWS
- Cluster locali: esegui il piano di controllo (control-plane) di Kubernetes e i nodi sull'Outpost.

Per entrambe le opzioni di implementazione, il piano di controllo (control-plane) di Kubernetes è completamente gestito da AWS. Puoi utilizzare le stesse API, gli stessi strumenti e le stesse console di Amazon EKS che utilizzi nel cloud per creare ed eseguire Amazon EKS su Outposts.

Il diagramma seguente illustra queste opzioni di implementazione.




## Quando utilizzare ciascuna opzione di implementazione

Sia i cluster locali sia quelli estesi sono opzioni di implementazione generiche e possono essere utilizzati per una vasta gamma di applicazioni.

Con i cluster estesi, puoi conservare la capacità del tuo Outpost perché il piano di controllo di Kubernetes funziona nella regione principale. AWS Questa opzione è adatta se disponi di una connettività di rete affidabile da Outpost alla regione. AWS Si consiglia di progettare le applicazioni in modo da garantire la stabilità statica mediante disconnessioni di rete dalla AWS regione. Il modo in cui Kubernetes gestisce le disconnessioni di rete tra il piano di controllo (control-plane) Kubernetes e i nodi potrebbe comportare tempi di inattività delle applicazioni. Per ulteriori informazioni sul

funzionamento di Kubernetes, consulta Scheduling, Preemption, and Eviction nella documentazione di Kubernetes.

Con i cluster locali puoi eseguire l'intero cluster Amazon EKS in locale su Outposts. Questa opzione consente di ridurre il rischio di tempi di inattività delle applicazioni che possono derivare da disconnessioni temporanee del cloud dalla rete. Queste disconnessioni di rete possono essere causate da interruzioni della fibra o da eventi meteorologici. Poiché l'intero cluster Amazon EKS viene eseguito in locale su Outposts, le applicazioni rimangono disponibili. Durante le disconnessioni del cloud dalla rete è possibile eseguire operazioni del cluster. Scegli i cluster locali se hai bisogno della continuità delle operazioni di Kubernetes tramite disconnessioni di rete o se richiesto dai mandati di residenza dei dati.

## Due implementazioni di cluster locali

Amazon EKS supporta due implementazioni di cluster locali sui rack Outposts. Un Outpost è configurato per utilizzare Amazon EBS o Amazon EC2 instance store come tipo di volume root per le istanze EC2 che vengono eseguite su di esso. L'implementazione utilizzata da Amazon EKS per il piano di controllo del cluster locale dipende da questa configurazione:

- AWS Outposts configurati con Amazon EBS. Il piano di controllo Kubernetes viene eseguito nel tuo AWS account sul tuo Outpost, con pila su tre istanze `etcd` Amazon EC2 supportate da Amazon EBS. Per ulteriori informazioni, consulta the section called "Esecuzione di cluster locali".
- AWS Outposts configurati con l'instance store EC2. Amazon EKS implementa un'architettura di cluster locali aggiornata. Il piano di controllo Kubernetes funziona su 6 istanze Amazon EC2 in un account di servizio AWS gestito su Outpost, con 3 istanze dedicate `etcd` e 3 istanze per gli altri componenti del piano di controllo. L'implementazione ha una maggiore parità con Amazon EKS nel cloud, segue il ciclo di vita della versione Amazon EKS Kubernetes e supporta funzionalità aggiuntive come componenti aggiuntivi e Bottlerocket-based nodi di lavoro di Amazon EKS (oltre a AL2023). È disponibile in tutte le AWS regioni standard in cui AWS Outposts è disponibile. Per ulteriori informazioni, consulta the section called "Esegui cluster locali (archivio di istanze EC2)".

Le funzionalità disponibili in ciascuna implementazione sono riepilogate nella tabella seguente.

## Confronto tra le opzioni di implementazione

La tabella seguente confronta le differenze tra i cluster estesi e le due implementazioni di cluster locali.

| Funzionalità | Cluster esteso | Cluster locale su Outposts con EBS | Cluster locale su Outposts con archivio di istanze EC2 |
|---|---|---|---|
| Posizione del piano di controllo Kubernetes | AWS Regione | Outpost | Outpost |
| Account del piano di controllo (control-plane) di Kubernetes | AWS conto | Il tuo AWS account | AWS-account gestito |
| Topologia del piano di controllo Kubernetes | Cloud-managed | 3 istanze Amazon EC2, in pila, ecc. | 6 istanze Amazon EC2 (`etcd`3+ 3 server API) |
| Storage sul piano di controllo | Cloud-managed | Amazon EBS | Archivio dell'istanza Amazon EC2 |
| Disponibilità regionale | Consulta la sezione Endpoint del servizio | Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (California settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Seoul), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda), Europa (Londra), Medio Oriente | Tutte le AWS regioni commerciali in cui AWS Outposts è disponibile |

| Funzionalità | Cluster esteso | Cluster locale su Outposts con EBS | Cluster locale su Outposts con archivio di istanze EC2 |
|---|---|---|---|
| | | (Bahrein) e Sud America (San Paolo) | |
| Versioni secondarie di Kubernetes | Versioni Amazon EKS supportate | Per informazioni, consultare the section called "Versioni della piattaforma EKS". | Versioni Amazon EKS supportate |
| Versioni della piattaforma | Scopri le versioni della piattaforma Amazon EKS | Per informazioni, consultare the section called "Versioni della piattaforma EKS". | Scopri le versioni della piattaforma Amazon EKS |
| Fattori di forma dell'Outpost | Rack dell'Outpost | Rack dell'Outpost | Rack dell'Outpost |
| Interfacce utente | Console di gestione AWS, AWS CLI, API Amazon EKS e `eksctl` AWS CloudFormation Terraform | Console di gestione AWS, AWS CLI, API Amazon EKS e `eksctl` AWS CloudFormation Terraform | Console di gestione AWS, AWS CLI, API Amazon EKS, AWS CloudFormation |
| Policy gestite | AmazonEKS ClusterPolicy e the section called "AWS policy gestita: AmazonEKS ServiceRolePolicy" | AmazonEKS LocalOutpostClusterPolicy e the section called "AWS policy gestita: AmazonEKS LocalOutpostServiceRolePolicy" | AmazonEKS ClusterPolicy e the section called "AWS policy gestita: AmazonEKS ServiceRolePolicy" |

| Funzionalità | Cluster esteso | Cluster locale su Outposts con EBS | Cluster locale su Outposts con archivio di istanze EC2 |
|---|---|---|---|
| VPC e sottoreti del cluster | Consulta la sezione the section called "Requisiti del VPC e delle sottoreti" | Per informazioni, consulta the section called "Creazione di un VPC e delle sottoreti" | Consulta la sezione the section called "Crea un VPC e delle sottoreti (store di istanze EC2)" |
| Accesso all'endpoint del cluster | Pubblico o privato o entrambi | Solo privati | Privato o pubblico e privato |
| Autenticazione del server API Kubernetes | AWS Identity and Access Management (IAM), OIDC e voci di accesso | IAM e certificati `x.509` | IAM, OIDC, voci di accesso e `x.509` certificati (per `aws-auth` ConfigMap le disconnessioni di rete) |
| Tipi di nodo | Self-managed solo | Self-managed solo | Self-managed solo |
| Tipi di elaborazione dei nodi | Amazon EC2 on demand | Amazon EC2 on demand | Amazon EC2 on demand |
| Tipi di archiviazione dei nodi | gp2 di Amazon EBS e SSD NVMe locali | gp2 di Amazon EBS e SSD NVMe locali | SSD NVMe locale |
| AMI ottimizzate per Amazon EKS | Amazon Linux, Windows e Bottlerocket | Amazon Linux | Amazon Linux e Bottlerocket |
| Versioni IP | Solo `IPv4` | Solo `IPv4` | Solo `IPv4` |
| Add-ons | Componenti aggiuntivi Amazon EKS o autogestiti | Self-managed componenti aggiuntivi | Componenti aggiuntivi Amazon EKS (elenco convalidato) o componenti aggiuntivi autogestiti |

| Funzionalità | Cluster esteso | Cluster locale su Outposts con EBS | Cluster locale su Outposts con archivio di istanze EC2 |
|---|---|---|---|
| Interfaccia di rete del container predefinita | Plug-in della CNI di Amazon VPC per Kubernetes | Plug-in della CNI di Amazon VPC per Kubernetes | Plug-in della CNI di Amazon VPC per Kubernetes |
| Registri del piano di controllo Kubernetes | CloudWatch Registri Amazon | CloudWatch Registri Amazon | CloudWatch Registri Amazon |
| Backup `etcd` | Cloud-managed | Customer-managed (`etcdctlo` backup di volumi Amazon EBS) | Gestito da Amazon EKS |
| Sistema di bilanciamento del carico | Utilizza AWS Load Balancer Controller per eseguire il provisioning solo di Application Load Balancer (nessun Network Load Balancer) | Utilizza AWS Load Balancer Controller per eseguire il provisioning solo di Application Load Balancer (nessun Network Load Balancer) | Utilizza AWS Load Balancer Controller per eseguire il provisioning solo di Application Load Balancer (nessun Network Load Balancer) |
| Crittografia a busta dei segreti | Supportato, vedi the section called "Abilitare la crittografia dei segreti" | Non supportata | Non supportata |
| IAM Roles for Service Accounts (IRSA) | Supportato, vedi the section called "Credenziali con IRSA" | Non supportata | Supportato (richiede la connettività alla AWS regione; non disponibile durante le disconnessioni di rete) |

| Funzionalità | Cluster esteso | Cluster locale su Outposts con EBS | Cluster locale su Outposts con archivio di istanze EC2 |
|---|---|---|---|
| EKS Pod Identity | Supportato, vedi the section called "Pod Identity" | Non supportata | Supportato (richiede la connettività alla AWS regione; non disponibile durante le disconnessioni di rete) |
| Risoluzione dei problemi | Consulta la sezione Risoluzione dei problemi | Per informazioni, consulta the section called "Risoluzione dei problemi dei cluster" | Consulta la sezione the section called "Risolvi i problemi relativi ai cluster (store di istanze EC2)" |

**Important**

- La creazione di un cluster locale su un RAM-shared Outpost pone il piano di controllo del cluster sotto l'accesso fisico del proprietario dell'Outpost
- Sull'instance store EC2 Outposts, IRSA ed EKS Pod Identity AWS dipendono da STS nella regione. AWS Durante le disconnessioni di rete, i carichi di lavoro che utilizzano IRSA o Pod Identity non possono ottenere nuove credenziali. Per ulteriori informazioni, consulta the section called "Preparati alle disconnessioni (archivio di istanze EC2)".

# Crea cluster Amazon EKS locali su AWS Outposts per l'alta disponibilità

Puoi utilizzare i cluster locali per eseguire l'intero cluster Amazon EKS localmente su AWS Outposts. Ciò consente di ridurre il rischio di tempi di inattività delle applicazioni che possono derivare da disconnessioni temporanee del cloud dalla rete. Queste disconnessioni di rete possono essere causate da interruzioni della fibra o da eventi meteorologici. Poiché l'intero cluster Kubernetes viene eseguito localmente su Outposts, le applicazioni rimangono disponibili. Durante le disconnessioni del cloud dalla rete è possibile eseguire operazioni del cluster. Per ulteriori

informazioni, consulta the section called "Preparazione per le disconnessioni". Il diagramma seguente mostra un'implementazione di un cluster locale.

I cluster locali sono generalmente disponibili per l'uso con i rack di Outposts.

## Supportata AWS Regioni

Puoi creare cluster locali AWS nelle seguenti regioni: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (California settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Seoul), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda), Europa (Londra), Medio Oriente (Bahrein) Bahrein) e Sud America (San Paolo). Per informazioni dettagliate sulle funzionalità supportate, consulta la sezione the section called "Confronto tra le opzioni di implementazione".

## Implementa un cluster Amazon EKS su AWS Outposts

Questo argomento offre una panoramica degli elementi da prendere in considerazione quando si esegue un cluster locale su un Outpost. L'argomento fornisce anche istruzioni su come implementare un cluster locale su un Outpost.

> **Important**
>
> - Queste considerazioni non sono trattate nella documentazione correlata di Amazon EKS. Se altri argomenti della documentazione di Amazon EKS sono in conflitto con le considerazioni qui riportate, segui le considerazioni qui.
> - Queste considerazioni sono soggette a modifiche e potrebbero cambiare frequentemente. Pertanto, ti consigliamo di rivedere regolarmente questo argomento.
> - Molte considerazioni sono diverse da quelle relative alla creazione di un cluster sul cloud. AWS

- I cluster locali supportano solo i rack dell'Outpost. Un singolo cluster locale può essere eseguito su più dell'Outpost fisici che comprendono un unico Outpost logico. Un singolo cluster locale non può essere eseguito su più Outpost logici. Ogni Outpost logico ha un singolo ARN dell'Outpost.
- I cluster locali eseguono e gestiscono il piano di controllo (control-plane) di Kubernetes nel tuo account sull'Outpost. Non puoi eseguire carichi di lavoro sulle istanze del piano di controllo (control-plane) di Kubernetes né modificare i relativi componenti. Questi nodi sono gestiti dal

servizio Amazon EKS. Le modifiche al piano di controllo (control-plane) di Kubernetes non persistono tra un'operazione di gestione automatica di Amazon EKS e l'altra, come l'applicazione di patch.

- I cluster locali supportano i componenti aggiuntivi autogestiti e i gruppi di nodi Amazon Linux autogestiti. I componenti aggiuntivi: plug-in per la Container Network Interface (CNI) di Amazon VPC per Kubernetes, kube-proxy e CoreDNS vengono installati automaticamente nei cluster locali.
- I cluster locali richiedono l'utilizzo di Amazon EBS su Outposts. Il tuo Outpost deve avere a disposizione Amazon EBS per l'archiviazione del piano di controllo (control-plane) di Kubernetes. Gli Outpost supportano solo i volumi `gp2` di Amazon EBS.
- I `PersistentVolumes` Kubernetes sostenuti da Amazon EBS sono supportati tramite il driver CSI per Amazon EBS.
- Le istanze del piano di controllo (control-plane) dei cluster locali sono configurate in una topologia in stack ad alta disponibilità. Due delle tre istanze del piano di controllo (control-plane) devono essere sempre integre per mantenere il quorum. Se il quorum viene perso, contatta l' AWS assistenza, poiché saranno necessarie alcune azioni sul lato del servizio per abilitare le nuove istanze gestite.

Prerequisiti

- Familiarità con le opzioni di distribuzione di Outposts, Select Instance Types e Placement Groups per i cluster Amazon EKS AWS su Outposts in base a considerazioni sulla capacità e requisiti e considerazioni VPC.
- Un Outpost esistente. Per ulteriori informazioni, consulta What is AWS Outposts.
- Lo strumento a riga di comando `kubectl` è installato sul computer o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` `apt-get`, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse

versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.

- Un principale IAM (utente o ruolo) con autorizzazioni `create` e `describe` per un cluster Amazon EKS. Per ulteriori informazioni, consultare the section called "Creazione di un cluster Kubernetes locale su un Outpost" e the section called "Elencare o descrivere tutti i cluster".

Quando viene creato un cluster Amazon EKS locale, il principale IAM che crea il cluster viene aggiunto in modo permanente. Il principale viene aggiunto specificamente alla tabella di autorizzazione RBAC Kubernetes come amministratore. Questa entità dispone di autorizzazioni `system:masters`. L'identità di questa entità non è visibile nella configurazione del cluster, per cui è fondamentale prendere nota dell'entità che ha creato il cluster, in modo da non eliminarla. Inizialmente, solo il principale che ha creato il server può effettuare chiamate al server API Kubernetes utilizzando `kubectl`. Se utilizzi la console per creare il cluster, assicurati che le stesse credenziali IAM siano presenti nella catena di credenziali AWS SDK quando `kubectl` esegui i comandi sul cluster. Dopo aver creato il cluster, è possibile concedere l'accesso ad altri principali IAM al cluster.

## Creazione di un cluster locale Amazon EKS.

Puoi creare un cluster locale con i seguenti strumenti descritti in questa pagina:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

Puoi anche utilizzare la AWS CLI, l'API Amazon EKS, gli AWS SDK AWS CloudFormationo Terraform per creare cluster su Outposts.

### `eksctl`

Per creare un cluster con **`eksctl`**

1. Installa la versione `0.215.0` o una versione successiva dello strumento da riga di `eksctl` comando sul tuo dispositivo o. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Copia i seguenti contenuti sul tuo dispositivo. Sostituisci i valori seguenti, quindi esegui il comando modificato per creare il file `outpost-control-plane.yaml`:
   - Sostituiscilo *`region-code`* con la AWS regione supportata in cui desideri creare il cluster.

- Sostituisci *my-cluster* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
- Sostituisci *vpc-ExampleID1* e *subnet-ExampleID1* con gli ID del tuo VPC e della sottorete esistenti. Il VPC e la sottorete devono soddisfare i requisiti in Crea un VPC e sottoreti per i cluster Amazon EKS su Outposts. AWS
- Sostituiscilo con *uniqueid* l'ID del tuo Outpost.
- Sostituisci *m5.large* con un tipo di istanza disponibile sul tuo Outpost. Prima di scegliere un tipo di istanza, consulta la sezione the section called "Considerazioni sulla capacità". Vengono implementate tre istanze del piano di controllo. Questo numero non può essere modificato.

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.36"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws: outposts:region-code:111122223333:outpost/op-
uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

Per un elenco completo di tutte le opzioni disponibili, consulta AWS Outposts Support e Config file schema nella documentazione di `eksctl`.

3. Crea il cluster utilizzando il file di configurazione creato nel passaggio precedente. `eksctl` crea un VPC e una sottorete sul tuo Outpost in cui implementare il cluster.

```
eksctl create cluster -f outpost-control-plane.yaml
```

Il provisioning del cluster richiede diversi minuti. Durante la creazione del cluster, vengono visualizzate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> Per visualizzare la maggior parte delle opzioni che è possibile specificare durante la creazione di un cluster con `eksctl`, utilizza il comando `eksctl create cluster --help`. Per visualizzare tutte le opzioni disponibili, puoi utilizzare un file `config`. Per ulteriori informazioni, consulta Uso dei file config e lo Schema dei file config nella documentazione di `eksctl`. Gli esempi di file di configurazione sono disponibili su GitHub.

Il comando `eksctl` ha creato automaticamente una voce di accesso per il principale IAM (utente o ruolo) che ha creato il cluster e ha concesso a quest'ultimo autorizzazioni di amministratore per gli oggetti Kubernetes sul cluster. Se non desideri che il creatore del cluster disponga di un accesso di amministratore agli oggetti Kubernetes sul cluster, aggiungi il seguente testo al file di configurazione precedente: `bootstrapClusterCreatorAdminPermissions: false` (allo stesso livello di `metadata`, `vpc` e `outpost`). Se hai aggiunto l'opzione, dopo la creazione del cluster devi creare una voce di accesso per almeno un principale IAM, altrimenti nessun principale IAM avrà accesso agli oggetti Kubernetes sul cluster.

Console di gestione AWS

Per creare il cluster con la Console di gestione AWS

1. Hai bisogno di un VPC e una sottorete esistenti che soddisfano i requisiti di Amazon EKS. Per ulteriori informazioni, consulta the section called "Creazione di un VPC e delle sottoreti".
2. Se disponi già di un ruolo IAM del cluster locale o intendi creare il cluster con `eksctl`, puoi ignorare questo passaggio. Per impostazione predefinita, `eksctl` crea un ruolo per te.
   a. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crea il ruolo IAM del cluster Amazon EKS. Per creare un ruolo IAM, è necessario assegnare l'azione `iam:CreateRole` (autorizzazione) al principale IAM che sta creando il ruolo.

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-
document file://"eks-local-cluster-role-trust-policy.json"
```

c. Allega al ruolo la policy gestita da Amazon EKS, denominata AmazonEKSLocalOutpostClusterPolicy. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

3. Aprire la Console Amazon EKS.
4. Nella parte superiore dello schermo della console, assicurati di aver selezionato una AWS regione supportata.
5. Seleziona Aggiungi cluster e quindi Crea.
6. Nella pagina Configure cluster (Configura cluster), inserisci o seleziona i valori per i campi seguenti:
   - Posizione del piano di controllo Kubernetes: scegli Outposts. AWS
   - Outpost ID (ID Outpost): scegli l'ID dell'Outpost su cui vuoi creare il tuo piano di controllo.
   - Instance type (Tipo di istanza): seleziona un tipo di istanza. Vengono visualizzati solo i tipi di istanza disponibili nel tuo Outpost. Nell'elenco a discesa, ogni tipo di istanza descrive per quanti nodi è consigliato il tipo di istanza. Prima di scegliere un tipo di istanza, consulta la sezione

the section called "Considerazioni sulla capacità". Tutte le repliche vengono implementate utilizzando lo stesso tipo di istanza. Non è possibile modificare il tipo di istanza dopo la creazione del cluster. Vengono implementate tre istanze del piano di controllo. Questo numero non può essere modificato.

- Nome: un nome per il cluster. Deve essere univoca nel tuo account. AWS Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
- Versione Kubernetes: scegli la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra utilizzare una versione precedente.
- Ruolo del servizio cluster: scegli il ruolo IAM del cluster Amazon EKS creato in un passaggio precedente per consentire al piano di controllo Kubernetes di gestire le risorse. AWS
- Accesso di amministratore del cluster Kubernetes: se desideri che il principale IAM (ruolo o utente) che crea il cluster disponga dell'accesso di amministratore agli oggetti Kubernetes sul cluster, accetta l'impostazione predefinita (Consenti). Amazon EKS crea una voce di accesso per il principale IAM e concede le autorizzazioni di amministratore del cluster per la voce di accesso. Per ulteriori informazioni sulle voci di accesso, consulta la sezione the section called "Voci di accesso".

  Se desideri che un principale IAM diverso da quello che crea il cluster disponga dell'accesso di amministratore agli oggetti del cluster Kubernetes, scegli l'opzione Non consentire. Dopo la creazione del cluster, qualsiasi principale IAM che dispone delle autorizzazioni IAM per creare voci di accesso può aggiungere voci di accesso per tutti i principali IAM che necessitano dell'accesso agli oggetti del cluster Kubernetes. Per ulteriori informazioni sulle autorizzazioni IAM richieste, consulta Actions defined by Amazon Elastic Kubernetes Service nella Documentazione di riferimento per l'autorizzazione ai servizi. Se scegli l'opzione Non consentire e non crei alcuna voce di accesso, nessun principale IAM avrà accesso agli oggetti Kubernetes sul cluster.
- Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Tagging delle risorse ". Quando hai finito con questa pagina, seleziona Avanti.

7. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:
   - VPC: scegli un VPC esistente. Il VPC deve disporre di un numero sufficiente di indirizzi IP da mettere a disposizione per il cluster, i nodi e le altre risorse Kubernetes da creare. Il VPC deve soddisfare i requisiti elencati in VPC requirements and considerations.

- Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. Le sottoreti scelte devono soddisfare i requisiti elencati in Subnet requirements and considerations.
- Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS. Amazon EKS crea automaticamente un gruppo di sicurezza che consente la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS. Se vuoi aggiungere i tuoi gruppi di sicurezza, non potrai modificare i gruppi scelti dopo la creazione del cluster. Per consentire agli host on-premises di comunicare con l'endpoint del cluster, è necessario consentire il traffico in entrata dal gruppo di sicurezza del cluster. Per i cluster che non dispongono di una connessione Internet in ingresso e in uscita (noti anche come cluster privati), devi eseguire una delle seguenti operazioni:
  - Aggiungi il gruppo di sicurezza associato agli endpoint VPC richiesti. Per ulteriori informazioni sugli endpoint richiesti, the section called "Utilizzo di endpoint VPC dell'interfaccia" consulta la sezione Accesso tramite sottorete ai servizi. AWS
  - Modifica il gruppo di sicurezza creato da Amazon EKS per consentire il traffico proveniente dal gruppo di sicurezza associato agli endpoint VPC. Quando hai finito con questa pagina, seleziona Avanti.

8. Nella pagina Configura osservabilità, puoi scegliere facoltativamente le opzioni Parametri e Registrazione del piano di controllo che desideri attivare. Per impostazione predefinita, i tipi di log sono disattivati.
   - Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Passaggio 1: attivazione delle metriche Prometheus".
   - Per ulteriori informazioni sulle opzioni Registrazione del piano di controllo, consulta the section called "Registrazione dei log del piano di controllo (control-plane)". Quando hai finito con questa pagina, seleziona Avanti.
9. Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE.

   Il provisioning del cluster richiede diversi minuti.

## Visualizzazione del cluster locale Amazon EKS

1. Dopo avere creato il cluster, puoi visualizzare le istanze del piano di controllo Amazon EC2 che sono state create.

   ```
   aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
   Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
   ```

   Di seguito viene riportato un output di esempio:

   ```
   "Name": "my-cluster-control-plane-id1"
   "Name": "my-cluster-control-plane-id2"
   "Name": "my-cluster-control-plane-id3"
   ```

   Ogni istanza è contaminata dal taint `node-role.eks-local.amazonaws.com/control-plane` per evitare che i carichi di lavoro vengano pianificati sulle istanze del piano di controllo. Per ulteriori informazioni sui taint, consulta Taints and Tolerations nella documentazione di Kubernetes. Amazon EKS monitora continuamente lo stato dei cluster locali. Eseguiamo operazioni di gestione automatiche, come applicazioni di patch di sicurezza e riparazioni di istanze non integre. Quando i cluster locali vengono disconnessi dal cloud, eseguiamo una serie di operazioni per garantire che il cluster venga ripristinato a uno stato di integrità al momento della riconnessione.
2. Se hai creato il cluster utilizzando `eksctl`, puoi ignorare questo passaggio. `eksctl` lo completerà automaticamente per tuo conto. Abilita `kubectl` per consentire la comunicazione con il cluster aggiungendo un nuovo contesto al file `config` `kubectl`. Per istruzioni su come creare e aggiornare il file, consulta the section called "Accedere al cluster con kubectl".

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Di seguito viene riportato un output di esempio.

   ```
   Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
   username/.kube/config
   ```
3. Per connetterti al server API Kubernetes del cluster locale, devi disporre dell'accesso al gateway locale per la sottorete o connetterti dall'interno del VPC. Per ulteriori informazioni sulla connessione di un rack Outpost alla rete locale, consulta Come funzionano i gateway locali per i rack nella Guida per l'utente di Outposts AWS . Se utilizzi l'instradamento VPC diretto e la

sottorete dell'Outpost ha un instradamento al gateway locale, gli indirizzi IP privati delle istanze del piano di controllo (control-plane) di Kubernetes vengono trasmessi automaticamente sulla rete locale. L'endpoint del server API Kubernetes del server locale è ospitato in Amazon Route 53 (Route 53). L'endpoint del servizio API può essere risolto dai server DNS pubblici negli indirizzi IP privati dei server dell'API Kubernetes.

Le istanze del piano di controllo (control-plane) di Kubernetes dei cluster locali sono configurate con interfacce di rete elastiche statiche con indirizzi IP privati fissi che non cambiano durante il ciclo di vita del cluster. Le macchine che interagiscono con il server API Kubernetes potrebbero non avere connettività a Route 53 durante le disconnessioni di rete. In tal caso, si consiglia di configurare `/etc/hosts` con gli indirizzi IP privati statici per continuare le operazioni. Ti consigliamo inoltre di configurare i server DNS locali e di collegarli al tuo Outpost. Per ulteriori informazioni, consulta la documentazione di AWS Outposts. Esegui il comando riportato per confermare che è stata stabilita la comunicazione con il cluster.

```
kubectl get svc
```

Di seguito viene riportato un output di esempio.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

4. (Facoltativo) Verifica l'autenticazione del cluster locale quando si trova in uno stato disconnesso dal cloud. AWS Per istruzioni, consulta the section called "Preparazione per le disconnessioni".

## Risorse interne

Amazon EKS crea le seguenti risorse sul tuo cluster. Le risorse sono destinate all'uso interno di Amazon EKS. Per garantire il corretto funzionamento del cluster, non modificare o alterare queste risorse.

- I seguenti pod mirror:
  - `aws-iam-authenticator-`*`node-hostname`*
  - `eks-certificates-controller-`*`node-hostname`*
  - `etcd-`*`node-hostname`*
  - `kube-apiserver-`*`node-hostname`*
  - `kube-controller-manager-`*`node-hostname`*

  - `kube-scheduler-`*`node-hostname`*
- I seguenti componenti aggiuntivi autogestiti:
  - `kube-system/coredns`
  - `kube-system/ kube-proxy` (non viene creato finché non aggiungi il primo nodo)
  - `kube-system/aws-node` (non viene creato finché non si aggiunge il primo nodo). I cluster locali utilizzano il plug-in per la CNI di Amazon VPC per Kubernetes per le reti di cluster. Non modificare la configurazione per le istanze del piano di controllo (pod denominati `aws-node-controlplane-*`). Ci sono variabili di configurazione che consentono di modificare il valore predefinito quando il plugin crea nuove interfacce di rete. Per ulteriori informazioni, consulta la documentazione su GitHub.
- I seguenti servizi:
  - `default/kubernetes`
  - `kube-system/kube-dns`
- Una `PodSecurityPolicy` denominata `eks.system`
- Una `ClusterRole` denominata `eks:system:podsecuritypolicy`
- Una `ClusterRoleBinding` denominata `eks:system`
- Oltre al gruppo di sicurezza del cluster, Amazon EKS crea un gruppo di sicurezza denominato nel tuo AWS account`eks-local-internal-do-not-use-or-edit-`*`cluster-name`*`-`*`uniqueid`* . Questo gruppo di sicurezza consente al traffico di fluire liberamente tra i componenti Kubernetes in esecuzione sulle istanze del piano di controllo (control-plane).

Fasi successive consigliate:

- Concedi al responsabile IAM che ha creato il cluster le autorizzazioni necessarie per visualizzare le risorse Kubernetes nel Console di gestione AWS
- Concedere alle entità IAM l'accesso al cluster. Per consentire alle entità di visualizzare le risorse Kubernetes nella console Amazon EKS, concedi le relative autorizzazioni richieste.
- Configurare la registrazione per il cluster
- Acquisisci familiarità con ciò che accade durante le disconnessioni dalla rete.
- Aggiungere nodi al cluster
- Prendi in considerazione la possibilità di impostare un piano di backup per `etcd`. Amazon EKS non supporta il backup e il ripristino automatici di `etcd` per i cluster locali. Per ulteriori informazioni, consulta Backing up an etcd cluster nella documentazione di Kubernetes. Le due opzioni principali

sono l'utilizzo di `etcdctl` per automatizzare l'acquisizione di snapshot o l'utilizzo del backup del volume di archiviazione di Amazon EBS.

## Scopri le versioni delle piattaforme Kubernetes e Amazon EKS per AWS Outposts

Le versioni della piattaforma cluster locale rappresentano le funzionalità del cluster Amazon EKS su Outposts. AWS Le versioni includono i componenti che vengono eseguiti sul piano di controllo (control-plane) Kubernetes, i cui flag del server API Kubernetes sono abilitati. Includono anche la versione corrente della patch di Kubernetes. Ogni versione secondaria Kubernetes dispone di una o più versioni della piattaforma associate. Le versioni della piattaforma per le diverse versioni secondarie di Kubernetes sono indipendenti. Le versioni della piattaforma per i cluster locali e i cluster Amazon EKS nel cloud sono indipendenti.

Quando è disponibile una nuova versione secondaria di Kubernetes per i cluster locali, ad esempio `1.31`, la versione della piattaforma iniziale per tale versione secondaria di Kubernetes inizia da `eks-local-outposts.1`. Tuttavia, Amazon EKS rilascia periodicamente nuove versioni della piattaforma per abilitare nuove impostazioni del piano di controllo Kubernetes e fornire soluzioni per problemi relativi alla sicurezza.

Quando nuove versioni della piattaforma del cluster locale diventano disponibili per una versione secondaria:

- Il numero di versione della piattaforma viene incrementato (`eks-local-outposts.n+1`).
- Amazon EKS aggiorna automaticamente tutti i cluster locali esistenti alla versione della piattaforma più recente per la versione secondaria di Kubernetes corrispondente. Gli aggiornamenti automatici delle versioni della piattaforma esistenti vengono implementati in modo incrementale. Il processo di rollout consiste nella sostituzione delle istanze gestite del piano di controllo (control-plane) Kubernetes in esecuzione su Outpost, una alla volta, fino a quando tutte le 3 non vengono sostituite da istanze nuove.
- Il processo di sostituzione delle istanze del piano di controllo (control-plane) Kubernetes verrà arrestato in caso di rischio di interruzione del servizio. Amazon EKS cercherà di sostituire un'istanza del piano di controllo (control-plane) Kubernetes solo nel caso in cui le altre 2 siano integre e soddisfino tutte le condizioni di disponibilità come nodo del cluster.
- Il completamento del rollout di una versione della piattaforma richiede in genere meno di 30 minuti. Se un cluster rimane attivo per `UPDATING` un periodo di tempo prolungato, consulta the section called "Risoluzione dei problemi dei cluster" e chiedi aiuto all'assistenza. AWS Non terminare mai

manualmente le istanze del piano di controllo di Kubernetes a meno che non venga richiesto dal supporto. AWS

- Amazon EKS potrebbe pubblicare un nuovo nodo AMI con una versione patch corrispondente. Tutte le versioni della patch sono compatibili tra il piano di controllo Kubernetes e le AMI dei nodi per una determinata versione secondaria di Kubernetes.

Le nuove versioni della piattaforma non introducono modifiche in conflitto né causano interruzioni del servizio.

I cluster locali vengono sempre creati con la versione della piattaforma disponibile più recente (`eks-local-outposts.n`) per la versione Kubernetes specificata.

Le versioni della piattaforma correnti e recenti sono descritte nelle tabelle qui di seguito.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/outposts/eks-outposts-platform-versions.adoc.atom
```

## Kubernetes versione 1.32

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma `1.32`: `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `ClusterTrustBundleAttest`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `PodSecurity`, `Priority`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| `1.32.12` | `eks-local-outposts.3` | Versione iniziale della versione Kubernete | 10 aprile 2026 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | s `v1.32` per i cluster Amazon EKS locali su Outposts. kube-proxy. `v1.32.12` AWS `v0.7.10`Autenticatore IAM. Plugin Amazon VPC CNI per Kubernetes. `v1.20.4` `v1.11.4`Core DNS. AWS Cloud Controller Manager. `v1.32.5` Versione Bottlerocket. `v1.57.0` | |

## Kubernetes versione 1.31

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma `1.31`: `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `ClusterTrustBundleAttest`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `PodSecurity`, `Priority`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| `1.31.14` | `eks-local-outposts.8` | Nuova versione della piattaforma con | 23 dicembre 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | correzioni e miglioramenti alla sicurezza. kube-proxy aggiornato a. `v1.31.14` AWS IAM Authenticator aggiornato a. `v0.7.8` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.4`. Versione Bottlerocket aggiornata a `v1.52.0`. | |
| `1.31.12` | `eks-local-outposts.5` | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. `v1.31.10` AWS IAM Authenticator aggiornato a. `v0.7.4` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.2`. Versione Bottlerocket aggiornata a `v1.47.0`. | 3 ottobre 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.9 | eks-local-outposts.4 | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. v1.31.9 AWS IAM Authenticator aggiornato a. v0.7.2 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.20.0. Versione Bottlerocket aggiornata a v1.43.0. | 9 agosto 2025 |
| 1.31.7 | eks-local-outposts.3 | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. v1.31.9 AWS IAM Authenticator aggiornato a. v0.7.1 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.5. Versione Bottlerocket aggiornata a v1.40.0. | 19 giugno 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.6 | eks-local-outposts.2 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione Bottlerocket aggiornata a v1.36.0. | 24 aprile 2025 |
| 1.31.6 | eks-local-outposts.1 | Rilascio iniziale di Kubernetes versione v1.31 per cluster Amazon EKS locali su Outposts. | 9 aprile 2025 |

## Kubernetes versione 1.30

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.30: CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, PodSecurity, Priority, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy e ValidatingAdmissionWebhook.

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.14 | eks-local-outposts.10 | Nuova versione della piattaforma con | 23 dicembre 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | correzioni e miglioramenti della sicurezza. AWS IAM Authenticator aggiornato a. `v0.7.8` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.4`. Versione Bottlerocket aggiornata a `v1.52.0`. | |
| `1.30.14` | `eks-local-outposts.7` | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. `v1.30.14` AWS IAM Authenticator aggiornato a. `v0.7.4` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.2`. Versione Bottlerocket aggiornata a `v1.47.0`. | 3 ottobre 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.13 | eks-local-outposts.6 | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. v1.30.13 AWS IAM Authenticator aggiornato a. v0.7.2 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.20.0. Versione Bottlerocket aggiornata a v1.43.0. | 09 agosto 2025 |
| 1.30.11 | eks-local-outposts.5 | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. v1.30.11 AWS IAM Authenticator aggiornato a. v0.7.1 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.5. Versione Bottlerocket aggiornata a v1.40.0. | 19 giugno 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.10 | eks-local-outposts.4 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione Bottlerocket aggiornata a v1.36.0. | 24 aprile 2025 |
| 1.30.10 | eks-local-outposts.3 | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. v1.30.10 AWS IAM Authenticator aggiornato a. v0.6.29 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.2. CoreDNS aggiornato a. v1.11.4 AWS Cloud Controller Manager aggiornato a. v1.30.8 Versione Bottlerocket aggiornata a a v1.34.0. | 27 marzo 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.7 | eks-local-outposts.2 | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. v1.30.7 AWS IAM Authenticator aggiornato a. v0.6.28 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.0. Versione di Bottlerocket aggiornata a v1.29.0. | 10 gennaio 2025 |
| 1.30.5 | eks-local-outposts.1 | Rilascio iniziale di Kubernetes versione v1.30 per cluster Amazon EKS locali su Outposts. | 13 novembre 2024 |

## Kubernetes versione 1.29

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.29: CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, PodSecurity, Priority, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection,

`TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| `1.29.15` | `eks-local-outposts.13` | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. AWS IAM Authenticator aggiornato a. `v0.7.8` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.4`. Versione Bottlerocket aggiornata a `v1.52.0`. | 23 dicembre 2025 |
| `1.29.15` | `eks-local-outposts.10` | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. AWS IAM Authenticator aggiornato a. `v0.7.4` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.2`. Versione Bottlerocket aggiornata a `v1.47.0`. | 3 ottobre 2025 |
| `1.29.15` | `eks-local-outposts.9` | Nuova versione della piattaforma con | 9 agosto 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | correzioni e miglioramenti della sicurezza. AWS IAM Authenticator aggiornato a. `v0.7.2` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.0`. Versione Bottlerocket aggiornata a `v1.43.0`. | |
| `1.29.15` | `eks-local-outposts.8` | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. `v1.29.15` AWS IAM Authenticator aggiornato a. `v0.7.1` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.19.5`. Versione Bottlerocket aggiornata a `v1.40.0`. | 19 giugno 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.29.14 | eks-local-outposts.7 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione Bottlerocket aggiornata a v1.36.0. | 24 marzo 2025 |
| 1.29.14 | eks-local-outposts.6 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. kube-proxy aggiornato a v1.29.14. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.2. CoreDNS aggiornato a. v1.11.4 AWS Cloud Controller Manager aggiornato a. v1.29.8 Versione Bottlerocket aggiornata a a v1.34.0. | 27 marzo 2025 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.29.11 | eks-local-outposts.5 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. kube-proxy aggiornato a v1.29.11. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.0. Immagine CoreDNS aggiornata a v1.11.3. Versione di Bottlerocket aggiornata a v1.29.0. | 10 gennaio 2025 |
| 1.29.9 | eks-local-outposts.4 | Nuova versione della piattaforma con correzioni e miglioramenti della sicurezza. kube-proxy aggiornato a. v1.29.9 AWS IAM Authenticator aggiornato a. v0.6.26 Versione di Bottlerocket aggiornata a v1.26.0. | 8 novembre 2024 |

| Versione di Kubernetes | Versione della piattaforma Amazon EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.29.6 | eks-local-outposts.3 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione di Bottlerocket aggiornata a v1.22.0. | 22 ottobre 2024 |
| 1.29.6 | eks-local-outposts.2 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione di Bottlerocket aggiornata a v1.21.0. | 27 agosto 2024 |
| 1.29.6 | eks-local-outposts.1 | Rilascio iniziale di Kubernetes versione v1.29 per cluster Amazon EKS locali su Outposts. | 20 agosto 2024 |

# Crea un VPC e sottoreti per i cluster Amazon EKS su AWS Outposts

Durante la creazione di un cluster locale, in genere si specificano un VPC e almeno una sottorete privata in esecuzione su Outposts. Questo argomento offre una panoramica dei requisiti e delle considerazioni relativi al VPC e alle sottoreti utilizzate con il cluster locale.

## Considerazioni e requisiti relativi al VPC

Durante la creazione di un cluster locale, il VPC specificato deve soddisfare i requisiti e le considerazioni seguenti:

- Assicurarsi che il VPC disponga di un numero sufficiente di indirizzi IP per il cluster locale, i nodi e le altre risorse Kubernetes da creare. Se il VPC da utilizzare non dispone di un numero sufficiente di indirizzi IP, aumentane il numero. Puoi farlo associando ulteriori blocchi Classless Inter-Domain Routing (CIDR) al tuo VPC. L'associazione di blocchi di CIDR privati (RFC 1918) e pubblici (non RFC 1918) può avvenire prima o dopo la creazione del cluster. Il riconoscimento del blocco CIDR associato a un VPC da parte del cluster può richiedere fino a cinque ore.
- Il VPC non può avere prefissi IP o intervalli CIDR IPv6 assegnati. A causa di queste limitazioni, le informazioni riportate alla pagina Assign more IP addresses to Amazon EKS nodes with prefixes e the section called "IPv6" non sono applicabili al VPC.
- Il VPC ha un nome host DNS e una risoluzione DNS abilitati. Senza queste funzionalità la creazione del cluster locale ha esito negativo, quindi dovrai abilitare le funzionalità e creare nuovamente il cluster. Per ulteriori informazioni, consulta Attributi DNS nel VPC nella Guida per l'utente di Amazon VPC.
- Per accedere al cluster locale tramite la rete locale, il VPC deve essere associato alla tabella di instradamento del gateway locale dell'Outpost. Per ulteriori informazioni, consulta le associazioni VPC nella Guida per l'utente di AWS Outposts.

## Considerazioni e requisiti relativi alle sottoreti

Durante la creazione del cluster, è necessario specificare almeno una sottorete privata. Se si specificano più sottoreti, le istanze del piano di controllo (control-plane) Kubernetes sono distribuite uniformemente tra le sottoreti. Se viene specificata più di una sottorete, le sottoreti devono essere presenti sullo stesso Outpost. Tuttavia, per comunicare tra loro le sottoreti devono avere gli instradamenti corretti e le autorizzazioni del gruppo di sicurezza. Quando crei un cluster locale, le sottoreti specificate devono soddisfare i requisiti seguenti:

- Le sottoreti si trovano tutte sullo stesso Outpost logico.
- Le sottoreti devono avere complessivamente almeno tre indirizzi IP disponibili per le istanze del piano di controllo (control-plane) Kubernetes. Se sono specificate tre sottoreti, ogni sottorete deve avere almeno un indirizzo IP disponibile. Se sono specificate due sottoreti, ogni sottorete deve avere almeno due indirizzi IP disponibili. Se è specificata una sottorete, la sottorete deve avere almeno tre indirizzi IP disponibili.
- Le sottoreti hanno un istradamento al gateway locale del rack dell'Outpost per accedere al server API Kubernetes sulla rete locale. Se le sottoreti non dispongono di un instradamento per il gateway locale del rack dell'Outpost, è necessario comunicare con il server API Kubernetes dall'interno del VPC.

- Le sottoreti devono utilizzare la denominazione basata sull'indirizzo IP. La denominazione basata sulle risorse di Amazon EC2 non è supportata da Amazon EKS.

## Accesso tramite sottorete a AWS services

Le sottoreti private del cluster locale su Outposts devono essere in grado di comunicare con i servizi regionali. AWS Ciò è possibile utilizzando un gateway NAT per l'accesso a Internet in uscita o, se si desidera mantenere privato tutto il traffico all'interno del VPC, tramite gli endpoint VPC di interfaccia.

### Utilizzo di un gateway NAT

Le sottoreti private del cluster locale su Outposts devono avere una tabella di routing associata con un instradamento a un gateway NAT in una sottorete pubblica che si trova nella zona di disponibilità parent dell'Outpost. La sottorete pubblica deve disporre di una route a un gateway Internet. Il gateway NAT consente l'accesso a Internet in uscita e impedisce le connessioni in entrata non richieste da Internet alle istanze sull'Outpost.

### Utilizzo di endpoint VPC dell'interfaccia

Se le sottoreti private del cluster locale su Outposts non dispongono di una connessione Internet in uscita o se si desidera mantenere tutto il traffico privato all'interno del VPC, è necessario creare i seguenti endpoint VPC ed endpoint gateway in una sottorete Regionale prima di creare il cluster.

| Endpoint | Tipo di endpoint |
| --- | --- |
| `com.amazonaws.` *`region-code`* `.ssm` | Interfaccia |
| `com.amazonaws.` *`region-code`* `.ssmmessages` | Interfaccia |
| `com.amazonaws.` *`region-code`* `.ec2messages` | Interfaccia |
| `com.amazonaws.` *`region-code`* `.ec2` | Interfaccia |
| `com.amazonaws.` *`region-code`* `.secretsmanager` | Interfaccia |
| `com.amazonaws.` *`region-code`* `.logs` | Interfaccia |

| Endpoint | Tipo di endpoint |
| --- | --- |
| `com.amazonaws.` *region-code* `.sts` | Interfaccia |
| `com.amazonaws.` *region-code* `.ecr.api` | Interfaccia |
| `com.amazonaws.` *region-code* `.ecr.dkr` | Interfaccia |
| `com.amazonaws.` *region-code* `.s3` | Gateway |

Gli endpoint devono soddisfare i seguenti requisiti:

- Creato in una sottorete privata situata nella zona di disponibilità parent dell'Outpost
- Abilitazione dei nomi DNS privati
- Disponi di un gruppo di sicurezza collegato che consente il traffico HTTPS in ingresso dall'intervallo CIDR della sottorete privata dell'Outpost.

La creazione di endpoint comporta dei costi. Per ulteriori informazioni, consultare Prezzi di AWS PrivateLink . Se i tuoi Pod devono accedere ad altri AWS servizi, devi creare endpoint aggiuntivi. Per un elenco completo degli endpoint, consulta i AWS servizi che si integrano con. AWS PrivateLink

## Crea un VPC

Puoi creare un VPC che soddisfi i requisiti precedenti utilizzando uno dei seguenti modelli: AWS CloudFormation

- Modello 1: questo modello crea un VPC con una sottorete privata su Outpost e una sottorete pubblica nella regione. AWS La sottorete privata ha un percorso verso Internet tramite un gateway NAT che risiede nella sottorete pubblica della regione. AWS Questo modello può essere utilizzato per creare un cluster locale in una sottorete con accesso a Internet in uscita.
- Modello 2: questo modello crea un VPC con una sottorete privata sull'Outpost e il set minimo di endpoint VPC necessari per creare un cluster locale in una sottorete che non dispone di accesso a Internet in ingresso o in uscita (nota anche come sottorete privata).

# Prepara cluster Amazon EKS locali su AWS Outposts per le disconnessioni di rete

Se la tua rete locale ha perso la connettività con il AWS cloud, puoi continuare a utilizzare il cluster Amazon EKS locale su un Outpost. Questo argomento illustra come preparare il cluster locale per le disconnessioni di rete e le considerazioni correlate.

- I cluster locali garantiscono stabilità e operatività continua durante le disconnessioni di rete temporanee e non pianificate. AWS Outposts rimane un'offerta completamente connessa che funge da estensione del AWS Cloud nel tuo data center. In caso di disconnessioni di rete tra Outpost e AWS Cloud, ti consigliamo di provare a ripristinare la connessione. Per istruzioni, consulta l'elenco di controllo per la risoluzione dei problemi della rete rack AWS Outposts nella Guida per l'utente di Outposts AWS . Per informazioni su come risolvere i problemi con i cluster locali, consulta the section called “Risoluzione dei problemi dei cluster”.
- Gli Outpost emettono un parametro `ConnectedStatus` che puoi utilizzare per monitorare lo stato di connettività del tuo Outpost. Per ulteriori informazioni, consulta Outposts Metrics nella Guida per l'utente di Outposts AWS .
- I cluster locali utilizzano IAM come meccanismo di autenticazione predefinito tramite AWS Identity and Access Management authenticator for Kubernetes. IAM non è disponibile durante le disconnessioni di rete. Pertanto, i cluster locali supportano un meccanismo di autenticazione alternativo basato sui certificati `x.509` che puoi utilizzare per connetterti al cluster durante le disconnessioni dalla rete. Per informazioni su come ottenere e utilizzare un certificato `x.509` per il tuo cluster, consulta the section called “Autenticazione al cluster locale durante una disconnessione dalla rete”.
- Se non riesci ad accedere a Route 53 durante le disconnessioni dalla rete, prendi in considerazione l'utilizzo dei server DNS locali nel tuo ambiente on-premises. Le istanze del piano di controllo (control-plane) di Kubernetes utilizzano indirizzi IP statici. Puoi configurare gli host che utilizzi per connetterti al cluster con il nome host e gli indirizzi IP dell'endpoint anziché utilizzare i server DNS locali. Per ulteriori informazioni, consulta DNS nella Guida per l'utente di AWS Outposts.
- Se prevedi un aumento del traffico delle applicazioni durante le disconnessioni dalla rete, puoi allocare capacità di elaborazione di riserva nel tuo cluster quando sei connesso al cloud. Le istanze Amazon EC2 sono incluse nel prezzo di Outposts. AWS Pertanto, l'esecuzione di istanze di riserva non influisce sui costi di utilizzo. AWS
- Durante le disconnessioni dalla rete, per consentire le operazioni di creazione, aggiornamento e dimensionamento dei carichi di lavoro, le immagini di container dell'applicazione devono

essere accessibili sulla rete locale e il cluster deve disporre di una capacità sufficiente. I cluster locali non ospitano un registro dei container per tuo conto. Se i pod in precedenza sono stati eseguiti su tali nodi, le immagini di container vengono memorizzate nella cache dei nodi. Se in genere estrai le immagini dei container dell'applicazione da Amazon ECR nel cloud, prendi in considerazione l'esecuzione di una cache o di un registro locale. Una cache o un registro locale è utile se hai bisogno di creare, aggiornare e dimensionare le risorse del carico di lavoro durante le disconnessioni dalla rete.

- I cluster locali utilizzano Amazon EBS come classe di archiviazione predefinita per i volumi persistenti e il driver Amazon EBS CSI per gestire il ciclo di vita dei volumi persistenti di Amazon EBS. Durante le disconnessioni di rete, i pod supportati da Amazon EBS non possono essere creati, aggiornati o dimensionati. Questo perché queste operazioni richiedono chiamate all'API Amazon EBS nel cloud. Se stai implementando carichi di lavoro stateful su cluster locali e hai bisogno di creare, aggiornare o dimensionare le operazioni durante le disconnessioni dalla rete, prendi in considerazione l'utilizzo di un meccanismo di archiviazione alternativo.
- Gli snapshot di Amazon EBS non possono essere creati o eliminati se AWS Outposts non può accedere alle API locali pertinenti ( AWS come le API per Amazon EBS o Amazon S3).
- Quando si integra ALB (Ingress) con AWS Certificate Manager (ACM), i certificati vengono inviati e archiviati nella memoria dell'istanza AWS Outposts ALB Compute. L'attuale terminazione TLS continuerà a funzionare in caso di disconnessione dalla regione. AWS La modifica delle operazioni in questo contesto non riuscirà (ad esempio nuove definizioni di ingresso, nuove operazioni dell'API su certificati ACM, dimensionamento del calcolo ALB o rotazione dei certificati). Per ulteriori informazioni, consulta Risoluzione dei problemi relativi al rinnovo gestito dei certificati nella Guida per l'utente di AWS Certificate Manager.
- I registri del piano di controllo (control-plane) di Amazon EKS vengono memorizzati nella cache locale sulle istanze del piano di controllo (control-plane) di Kubernetes durante le disconnessioni dalla rete. Al momento della riconnessione, i registri vengono inviati ai CloudWatch registri nella regione principale. AWS Per monitorare il cluster in locale utilizzando l'endpoint dei parametri del server API Kubernetes o Fluent Bit per i log, puoi utilizzare Prometheus, Grafana e le soluzioni partner di Amazon EKS.
- Se utilizzi il AWS Load Balancer Controller su Outposts per il traffico delle applicazioni, i Pod esistenti gestiti dal Load AWS Balancer Controller continuano a ricevere traffico durante le disconnessioni di rete. I nuovi Pod creati durante le disconnessioni di rete non ricevono traffico finché Outpost non viene ricollegato al Cloud. AWS Valuta la possibilità di impostare il numero di repliche per le applicazioni connesse al AWS cloud per soddisfare le tue esigenze di scalabilità durante le disconnessioni di rete.

- Il plug-in della CNI di Amazon VPC per Kubernetes utilizza per impostazione predefinita la modalità IP secondaria. È configurato con `WARM_ENI_TARGET=1`, che consente al plug-in di mantenere disponibile "un'interfaccia di rete completamente elastica" degli indirizzi IP disponibili. Puoi modificare i valori `WARM_ENI_TARGET`, `WARM_IP_TARGET` e `MINIMUM_IP_TARGET` in base alle tue esigenze di scalabilità durante uno stato di disconnessione. Per ulteriori informazioni, consultate il file readme relativo al plugin. GitHub Per un elenco del numero massimo di Pod supportato da ogni tipo di istanza, consultate il file eni-max-pods.txt su. GitHub

## Ottimizza il comportamento di failover dei pod Kubernetes

Durante le disconnessioni di rete, il controller del ciclo di vita del nodo Kubernetes contrassegna i nodi irraggiungibili con l'effetto di contaminazione. `node.kubernetes.io/unreachable` `NoExecute` Per impostazione predefinita, i pod senza una tolleranza corrispondente vengono eliminati dopo 300 secondi (5 minuti). È possibile regolare questo comportamento utilizzando DaemonSets, configurandolo, `tolerationSeconds` su pod applicativi o implementando un controller personalizzato, in modo da consentire ai pod di rimanere sui propri nodi durante le disconnessioni temporanee senza sfratti inutili. Per indicazioni ed esempi dettagliati, consulta https://docs.aws.amazon.com/eks/latest/best-practices/hybrid-nodes-network-disconnection-best-practices.html # tune_kubernetes_pod_failover_behavior [Tune Kubernetes pod failover behavior] nella _Amazon EKS Best Practices Guide.

## Autenticazione al cluster locale durante una disconnessione dalla rete

AWS Identity and Access Management (IAM) non è disponibile durante le disconnessioni di rete. Non puoi autenticarti nel cluster locale utilizzando le credenziali IAM durante una disconnessione. Tuttavia, durante la disconnessione, potrai connetterti al cluster tramite la rete locale utilizzando i certificati `x509`. È necessario scaricare e archiviare un certificato `X509` client da utilizzare durante le disconnessioni. In questo argomento viene illustrato come creare e utilizzare il certificato per autenticarsi nel cluster quando è disconnesso.

1. Creare una richiesta di firma del certificato.
   a. Genera una richiesta di firma del certificato.

```
openssl req -new -newkey rsa:4096 -nodes -days 365 \
    -keyout admin.key -out admin.csr -subj "/CN=admin"
```

   b. Crea una richiesta di firma del certificato in Kubernetes.

```
BASE64_CSR=$(cat admin.csr | base64 -w 0)
cat << EOF > admin-csr.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
  - client auth
EOF
```

2. Crea una richiesta di firma del certificato utilizzando `kubectl`.

```
kubectl create -f admin-csr.yaml
```

3. Controlla lo stato della richiesta di firma del certificato.

```
kubectl get csr admin-csr
```

   Di seguito viene riportato un output di esempio.

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

   Kubernetes ha creato la richiesta di firma del certificato.

4. Approva la richiesta di firma del certificato.

```
kubectl certificate approve admin-csr
```

5. Controlla nuovamente lo stato della richiesta di firma del certificato per l'approvazione.

```
kubectl get csr admin-csr
```

   Di seguito viene riportato un output di esempio:

```
NAME       AGE   REQUESTOR                    CONDITION
admin-csr  11m   kubernetes-admin             Approved
```

6. Recupera e verifica il certificato.

   a. Recupera il certificato.

   ```
   kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode >
    admin.crt
   ```

   b. Verifica il certificato.

   ```
   cat admin.crt
   ```

7. Crea un'associazione di ruoli del cluster per un utente `admin`.

   ```
   kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
       --user=admin --group=system:masters
   ```

8. Genera un kubeconfig con ambito utente per uno stato disconnesso.

   Puoi generare un file `kubeconfig` utilizzando il certificati `admin` scaricati. Sostituisci *`my-cluster`* e *`apiserver-endpoint`* nei seguenti comandi.

   ```
   aws eks describe-cluster --name my-cluster \
       --query "cluster.certificateAuthority" \
       --output text | base64 --decode > ca.crt
   ```

   ```
   kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
       --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
   ```

   ```
   kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
       --client-certificate=admin.crt --client-key=admin.key --embed-certs
   ```

   ```
   kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
       --cluster my-cluster --user admin
   ```

   ```
   kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
   ```

9. Visualizza il file `kubeconfig`.

   ```
   kubectl get nodes --kubeconfig admin.kubeconfig
   ```

10. Se disponi di servizi già in produzione sul tuo Outpost, salta questo passaggio. Se Amazon EKS è l'unico servizio in esecuzione sul tuo Outpost e quest'ultimo non è attualmente in produzione, puoi simulare una disconnessione di rete. Prima di entrare in produzione con il cluster locale, puoi simulare una disconnessione per assicurarti di riuscire ad accedere al cluster quando è in uno stato disconnesso.
    a. Applica le regole del firewall ai dispositivi di rete che collegano Outpost alla AWS regione. Questo disconnette il link del servizio dell'Outpost. Non puoi creare nuove istanze. Le istanze attualmente in esecuzione perdono la connettività alla AWS regione e a Internet.
    b. Puoi testare la connessione al cluster locale durante una disconnessione tramite il certificato `x509`. Assicurati di modificare la `kubeconfig` con il `admin.kubeconfig` che hai creato in una fase precedente. Sostituiscilo *`my-cluster`* con il nome del cluster locale.

        ```
        kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
        ```

    Se riscontri problemi con i cluster locali mentre sono disconnessi, ti consigliamo di inviare una richiesta di supporto.

## Seleziona i tipi di istanze e i gruppi di posizionamento per i cluster Amazon EKS su AWS Outposts basati su considerazioni sulla capacità

Questo argomento fornisce assistenza per la selezione del tipo di istanza del piano di controllo (control-plane) di Kubernetes e (facoltativamente) per l'uso di gruppi di posizionamento per soddisfare requisiti di alta disponibilità per il tuo cluster Amazon EKS locale su un Outpost.

Prima di selezionare un tipo di istanza (`m5`, `c5` o `r5`) da utilizzare per il piano di controllo (control-plane) di Kubernetes del cluster locale su Outposts, devi verificare i tipi di istanza disponibili nella configurazione dell'Outpost. Dopo aver identificato i tipi di istanza disponibili, devi selezionare la dimensione dell'istanza (`large`, `xlarge` o `2xlarge`) in base al numero di nodi richiesti dai carichi di lavoro. La tabella seguente fornisce consigli per la selezione della dimensione dell'istanza.

> **Note**
>
> Le dimensioni delle istanze devono essere assegnate ai tuoi Outposts. Assicurati di disporre di una capacità sufficiente per tre istanze della dimensione disponibile negli Outposts per tutta la durata del cluster locale. Per un elenco dei tipi di istanza Amazon EC2 disponibili, consulta le sezioni relative al calcolo e all'archiviazione in Funzionalità dei rack AWS Outposts.

| Numero di nodi | Dimensione dell'istanza del piano di controllo (control-plane) Kubernetes |
|---|---|
| 1-20 | `large` |
| 21-100 | `xlarge` |
| 101-250 | `2xlarge` |
| 251-500 | `4xlarge` |

L'archiviazione per il piano di controllo (control-plane) di Kubernetes richiede 246 GB di spazio di archiviazione Amazon EBS per cluster locale per soddisfare gli IOPS richiesti per `etcd`. Quando viene creato il cluster, i volumi Amazon EBS vengono allocati automaticamente per tuo conto.

## Collocamento del piano di controllo (control-plane)

Se non specifichi un gruppo di posizionamento con la proprietà `OutpostConfig.ControlPlanePlacement.GroupName`, le istanze Amazon EC2 fornite per il tuo piano di controllo (control-plane) di Kubernetes non ricevono alcuna specifica applicazione del collocamento dell'hardware nella capacità sottostante disponibile sul tuo Outpost.

Puoi utilizzare gruppi di collocamento per soddisfare i requisiti di alta disponibilità del tuo cluster Amazon EKS locale su un Outpost. Specificando un gruppo di posizionamento durante la creazione del cluster, influisci sul collocamento delle istanze del piano di controllo (control-plane) di Kubernetes. Le istanze sono distribuite su hardware sottostante indipendente (rack o host), riducendo al minimo l'impatto delle istanze correlate in caso di guasti hardware.

Il tipo di spread che puoi configurare dipende dal numero di rack Outpost esistenti nella tua implementazione.

- Implementazioni con uno o due rack fisici in un unico Outpost logico: occorrono almeno tre host configurati con il tipo di istanza scelto per le tue istanze del piano di controllo (control plane) di Kubernetes. Un gruppo di posizionamento degli spread che utilizza lo spread a livello di host garantisce che tutte le istanze del piano di controllo (control-plane) di Kubernetes vengano eseguite su host distinti all'interno dei rack sottostanti disponibili nella tua implementazione Outpost.
- Implementazioni con tre o più rack fisici in un unico Outpost logico: occorrono almeno tre host configurati con il tipo di istanza scelto per le tue istanze del piano di controllo (control plane) di

Kubernetes. Un gruppo di posizionamento degli spread che utilizza lo spread a livello di rack garantisce che tutte le istanze del piano di controllo (control-plane) di Kubernetes vengano eseguite su rack distinti nella tua implementazione Outpost. In alternativa, puoi utilizzare il gruppo di collocamento spread a livello di host come descritto nell'opzione precedente.

Sei responsabile della creazione del gruppo di collocamento desiderato. Specifichi il gruppo di collocamento quando richiami l'API `CreateCluster`. Per ulteriori informazioni, sui gruppi di posizionamento e per le istruzioni per crearli, consulta Placement Groups nella Guida per l'utente di Amazon EC2.

- Quando viene specificato un gruppo di collocamento, deve essere disponibile una capacità slot sull'Outpost per creare correttamente un cluster Amazon EKS locale. La capacità varia a seconda se utilizzi il tipo di spread host o rack. Se la capacità è insufficiente, il cluster rimane nello stato `Creating`. Puoi controllare `Insufficient Capacity Error` nel campo di integrità della risposta dell'API DescribeCluster. Devi liberare capacità per l'avanzamento del processo di creazione.
- Durante gli aggiornamenti della piattaforma e della versione del cluster locale Amazon EKS, le istanze del piano di controllo (control-plane) di Kubernetes dal tuo cluster vengono sostituite da nuove istanze utilizzando una strategia di aggiornamento in sequenza. Durante questo processo di sostituzione, ogni istanza del piano di controllo (control plane) termina, liberando il rispettivo slot. Al suo posto viene fornita una nuova istanza aggiornata. L'istanza aggiornata potrebbe essere collocata nello slot che è stato rilasciato. Se lo slot viene utilizzato da un'altra istanza non correlata e la capacità rimanente non soddisfa i requisiti di topologia spread richiesti, il cluster rimane nello stato `Updating`. Puoi controllare `Insufficient Capacity Error` rispettivo nel campo di integrità della risposta dell'API DescribeCluster. Devi liberare la capacità per l'avanzamento del processo di aggiornamento e la ridefinizione dei livelli di alta disponibilità precedenti.
- È possibile creare un massimo di 500 gruppi di collocamento per account in ciascuna regione. AWS Per ulteriori informazioni, consulta General rules and limitations nella Guida per l'utente di Amazon EC2.

## Risolvi i problemi relativi ai cluster Amazon EKS locali su AWS Outposts

Questo argomento illustra alcuni errori comuni che si potrebbero verificare durante l'utilizzo dei cluster locali e il modo in cui risolverli. I cluster locali siano simili ai cluster Amazon EKS sul cloud, ma esistono delle differenze nel modo in cui sono gestiti da Amazon EKS.

> **Important**
>
> Non terminare mai alcuna istanza gestita del `Kubernetes` piano di controllo del cluster locale EKS in esecuzione su Outpost a meno che non sia esplicitamente indicato dall'assistenza. AWS La chiusura di queste istanze comporta un rischio per la disponibilità del servizio di cluster locale, inclusa la perdita del cluster locale nel caso in cui più istanze vengano terminate contemporaneamente. Le istanze del piano di controllo (control-plane) `Kubernetes` del cluster locale EKS sono identificate dal tag `eks-local:controlplane-name` sulla console dell'istanza EC2.

Funzionamento dell'API

I cluster locali vengono creati tramite l'API Amazon EKS, ma vengono eseguiti in modo asincrono. Ciò significa che le richieste all'API Amazon EKS vengono restituite immediatamente per i cluster locali. Tuttavia, queste richieste potrebbero avere esito positivo, anticipare l'errore a causa di errori di convalida degli input oppure fallire e riportare errori di convalida descrittivi. Questo funzionamento è simile a quello dell'API Kubernetes.

I cluster locali non effettuano la transizione a uno stato `FAILED`. Amazon EKS prova a riconciliare lo stato del cluster con lo stato desiderato richiesto dall'utente in modo continuo. Di conseguenza, un cluster locale può rimanere a lungo nello stato `CREATING`, fino a quando il problema di base non viene risolto.

Descrizione del campo di integrità del cluster

I problemi del cluster locale possono essere scoperti utilizzando il comando Amazon EKS CLI https://docs.aws.amazon.com/cli/latest/reference/eks/describe-cluster.html describe-cluster. AWS I problemi relativi ai cluster locali vengono evidenziati dal campo `cluster.health` della risposta del comando `describe-cluster`. Il messaggio contenuto in questo campo include un codice di errore, un messaggio descrittivo e gli ID delle risorse correlate. Queste informazioni sono disponibili solo tramite l'API e la CLI di Amazon EKS. AWS Nell'esempio seguente, sostituiscilo *`my-cluster`* con il nome del tuo cluster locale.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

Di seguito viene riportato un output di esempio.

```
{
```

```
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
        }
    ]
}
```

Se il problema non è risolvibile, potrebbe essere necessario eliminare il cluster locale e crearne uno nuovo. Ad esempio, ciò potrebbe verificarsi nel caso in cui si provi ad allocare un cluster con un tipo di istanza che non è disponibile sul tuo Outpost. La tabella seguente include gli errori più comuni relativi allo stato di integrità.

| Scenario di errore | Codice | Messaggio | ResourceIds |
|---|---|---|---|
| Non è stato possibile trovare le sottoreti fornite. | `ResourceNotFound` | `The subnet ID` *`subnet-id`* `does not exist` | Tutti gli ID di sottorete forniti |
| Le sottoreti fornite non appartengono allo stesso VPC. | `ConfigurationConflict` | `Subnets specified must belong to the same VPC` | Tutti gli ID di sottorete forniti |
| Alcune delle sottoreti fornite non appartengono all'Outpost specificato. | `ConfigurationConflict` | `Subnet` *`subnet-id`* `expected to be in` *`outpost-arn`* `, but is in` *`other-outpost-arn`* | ID della sottorete problematico |
| Alcune delle sottoreti fornite non appartengono ad alcun Outpost. | `ConfigurationConflict` | `Subnet` *`subnet-id`* `is not part of any Outpost` | ID della sottorete problematico |
| Alcune sottoreti fornite non dispongono di | `ResourceLimitExceeded` | `The specified subnet does` | ID della sottorete problematico |

| Scenario di errore | Codice | Messaggio | ResourceIds |
| --- | --- | --- | --- |
| indirizzi liberi sufficienti per creare interfacce di rete elastiche per le istanze del piano di controllo (control-plane). | | `not have enough free addresses to satisfy the request.` | |
| Il tipo di istanza del piano di controllo (control-plane) specificato non è supportato sul tuo Outpost. | `ConfigurationConflict` | `The instance type` *`type`* `is not supported in Outpost` *`outpost-arn`* | ARN del cluster |
| Hai terminato un'istanza Amazon EC2 del piano di controllo oppure `run-instance` ha avuto esito positivo ma sono state riscontrate modifiche a `Terminated` . Ciò può verificarsi per un periodo di tempo dopo la riconnessione dell'Outpost e gli errori interni di Amazon EBS causano il fallimento di un flusso di lavoro interno di Amazon EC2. | `InternalFailure` | `EC2 instance state "Terminated" is unexpected` | ARN del cluster |

| Scenario di errore | Codice | Messaggio | ResourceIds |
| --- | --- | --- | --- |
| Capacità insufficiente nell'Outpost. Ciò può verificarsi anche quando viene creato un cluster se un Outpost viene disconnesso dalla AWS regione. | `ResourceLimitExceeded` | `There is not enough capacity on the Outpost to launch or start the instance.` | ARN del cluster |
| Il tuo account supera la quota del gruppo di sicurezza | `ResourceLimitExceeded` | Messaggio di errore restituito dall'API Amazon EC2 | ID del VPC di destinazione |
| Il tuo account supera la quota dell'interfaccia di rete elastica | `ResourceLimitExceeded` | Messaggio di errore restituito dall'API Amazon EC2 | ID della sottorete di destinazione |
| Le istanze del piano di controllo non erano raggiungibili tramite Systems Manager. AWS Per la risoluzione, consulta la sezione the section called "Le istanze del piano di controllo non sono raggiungibili tramite AWS Systems Manager". | `ClusterUnreachable` | Le istanze del piano di controllo di Amazon EKS non sono raggiungibili tramite SSM. Verifica la configurazione SSM e di rete e fai riferimento alla documentazione per la risoluzione dei problemi di EKS su Outposts. | ID di istanza Amazon EC2 |

| Scenario di errore | Codice | Messaggio | ResourceIds |
|---|---|---|---|
| Si è verificato un errore durante la raccolta dei dettagli per un gruppo di sicurezza gestito o un'interfaccia di rete elastica. | In base al codice di errore del client Amazon EC2. | Messaggio di errore restituito dall'API Amazon EC2 | Tutti gli ID dei gruppi di sicurezza gestiti |
| Si è verificato un errore durante l'autorizzazione o la revoca delle regole di ingresso dei gruppi di sicurezza. Questo vale per i gruppi di sicurezza sia del cluster sia del piano di controllo. | In base al codice di errore del client Amazon EC2. | Messaggio di errore restituito dall'API Amazon EC2 | ID gruppo di sicurezza problematico |
| Si è verificato un errore durante l'eliminazione di un'interfaccia di rete elastica per un'istanza del piano di controllo | In base al codice di errore del client Amazon EC2. | Messaggio di errore restituito dall'API Amazon EC2 | ID dell'interfaccia di rete elastica problematica |

La tabella seguente elenca gli errori di altri AWS servizi presentati nel campo di integrità della risposta. `describe-cluster`

| Codice di errore Amazon EC2 | Codice del problema di integrità del cluster | Description |
|---|---|---|
| `AuthFailure` | `AccessDenied` | Questo problema può verificar si per una serie di motivi. Di solito si verificano se un |

| Codice di errore Amazon EC2 | Codice del problema di integrità del cluster | Description |
|---|---|---|
| | | tag utilizzato dal servizio per definire la policy dei ruoli collegati al servizio viene rimosso accidentalmente dal piano di controllo. In tal caso, Amazon EKS non è più in grado di gestire e monitorare queste AWS risorse. |
| `UnauthorizedOperation` | `AccessDenied` | Questo problema può verificarsi per una serie di motivi. Di solito si verificano se un tag utilizzato dal servizio per definire la policy dei ruoli collegati al servizio viene rimosso accidentalmente dal piano di controllo. In tal caso, Amazon EKS non è più in grado di gestire e monitorare queste AWS risorse. |
| `InvalidSubnetID.NotFound` | `ResourceNotFound` | Questo errore si verifica quando non viene trovato l'ID di sottorete per le regole di ingresso di un gruppo di sicurezza. |
| `InvalidPermission.NotFound` | `ResourceNotFound` | Questo errore si verifica quando le autorizzazioni per le regole di ingresso di un gruppo di sicurezza non sono corrette. |

| Codice di errore Amazon EC2 | Codice del problema di integrità del cluster | Description |
|---|---|---|
| `InvalidGroup.NotFound` | `ResourceNotFound` | Questo errore si verifica quando non viene trovato il gruppo delle regole di ingresso di un gruppo di sicurezza. |
| `InvalidNetworkInterfaceID.NotFound` | `ResourceNotFound` | Questo errore si verifica quando non viene trovato l'ID dell'interfaccia di rete per le regole di ingresso di un gruppo di sicurezza. |
| `InsufficientFreeAddressesInSubnet` | `ResourceLimitExceeded` | Questo errore si verifica quando viene superata la quota di risorse della sottorete. |
| `InsufficientCapacityOnOutpost` | `ResourceLimitExceeded` | Questo errore si verifica quando viene superata la quota di capacità dell'outpost. |
| `NetworkInterfaceLimitExceeded` | `ResourceLimitExceeded` | Questo errore si verifica quando viene superata la quota dell'interfaccia di rete elastica. |
| `SecurityGroupLimitExceeded` | `ResourceLimitExceeded` | Questo errore si verifica quando viene superata la quota del gruppo di sicurezza. |

| Codice di errore Amazon EC2 | Codice del problema di integrità del cluster | Description |
|---|---|---|
| `VcpuLimitExceeded` | `ResourceLimitExceeded` | Ciò si verifica quando si crea un'istanza Amazon EC2 in un nuovo account. L'errore potrebbe essere simile al seguente: "`You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit.`" |
| `InvalidParameterValue` | `ConfigurationConflict` | Amazon EC2 restituisce questo codice di errore se il tipo di istanza specificato non è supportato nell'Outpost. |
| Tutti gli altri errori | `InternalFailure` | Nessuno |

Impossibile creare o modificare i cluster

I cluster locali richiedono autorizzazioni e policy diverse rispetto ai cluster Amazon EKS ospitati nel cloud. Quando un cluster non riesce a creare e genera un `InvalidPermissions` errore, ricontrolla che al ruolo del cluster che stai utilizzando sia associata la policy AmazonEKSLocalOutpostClusterPolicy gestita. Tutte le altre chiamate API richiedono lo stesso set di autorizzazioni dei cluster Amazon EKS sul cloud.

### Il cluster è bloccato nello **`stato`** CREATING

La quantità di tempo necessaria per creare un cluster locale varia a seconda di diversi fattori. Questi fattori includono la configurazione della rete, dell'Outpost e del cluster. In generale, viene creato un cluster locale che passa allo stato `ACTIVE` entro 15-20 minuti. Se un cluster locale rimane nello stato `CREATING`, puoi richiamare `describe-cluster` per informazioni sulla causa nel campo di output `cluster.health`.

I problemi più comuni sono i seguenti:

- Il cluster non può connettersi all'istanza del piano di controllo dalla AWS regione in cui si trova Systems Manager. Puoi eseguire una verifica chiamando `aws ssm start-session --target` *`instance-id`* da un host bastione nella regione. Se il comando non funziona, controlla se Systems Manager è in esecuzione sull'istanza del piano di controllo (control-plane). Oppure, un'altra soluzione alternativa consiste nell'eliminare il cluster e quindi ricrearlo.
- Le istanze del piano di controllo (control-plane) non vengono create a causa delle autorizzazioni della chiave KMS per i volumi EBS. Con le chiavi KMS gestite dall'utente per i volumi EBS crittografati, le istanze del piano di controllo (control-plane) termineranno se la chiave non è accessibile. Se le istanze vengono terminate, passa a una chiave KMS AWS gestita o assicurati che la policy relativa alle chiavi gestite dall'utente conceda le autorizzazioni necessarie per il ruolo del cluster.
- Le istanze del piano di controllo di Systems Manager potrebbero non avere accesso a Internet. Verifica se la sottorete fornita durante la creazione del cluster dispone di un gateway NAT e un VPC con un gateway Internet. Usa VPC Reachability Analyzer per verificare se l'istanza del control plane può raggiungere il gateway Internet. Per ulteriori informazioni, consulta la sezione Getting started with VPC Reachability Analyzer (Nozioni di base su VPC Reachability Analyzer).
- L'ARN del ruolo fornito è privo di alcune policy. Controlla se la policy AWS gestita: è AmazonEKSLocalOutpostClusterPolicy stata rimossa dal ruolo. Ciò può verificarsi anche se uno AWS CloudFormation stack non è configurato correttamente.
- Tutte le sottoreti fornite devono essere associate allo stesso Outpost ed essere in grado di raggiungersi tra loro. Quando vengono specificate più sottoreti durante la creazione del cluster, Amazon EKS prova a distribuire le istanze del piano di controllo su più sottoreti.
- I gruppi di sicurezza gestiti da Amazon EKS vengono applicati all'interfaccia di rete elastica. Tuttavia, altri elementi di configurazione, come le regole del firewall NACL, potrebbero essere in conflitto con le regole per l'interfaccia di rete elastica.

- La configurazione del VPC e del DNS di sottorete è configurata in modo errato o mancante. Consulta Crea un VPC e sottoreti per i cluster Amazon EKS su Outposts. AWS

Il cluster è bloccato nello stato UPDATING

Amazon EKS aggiorna automaticamente tutti i cluster locali esistenti alla versione della piattaforma più recente per la versione secondaria di Kubernetes corrispondente. Per ulteriori informazioni sulle versioni della piattaforma, consulta the section called "Versioni della piattaforma EKS".

Durante l'implementazione automatica di una versione della piattaforma, lo stato di un cluster cambia in `UPDATING`. Il processo di aggiornamento consiste nella sostituzione di tutte le istanze del piano di controllo di Kubernetes con nuove istanze contenenti le ultime patch di sicurezza e correzioni di bug rilasciate per la rispettiva versione secondaria di Kubernetes. In generale, il processo di aggiornamento della piattaforma di un cluster locale viene completato in meno di 30 minuti e il cluster torna allo stato `ACTIVE`. Se un cluster locale rimane nello stato `UPDATING` per un periodo di tempo prolungato, puoi chiamare `describe-cluster` per controllare le informazioni sulla causa nel campo di output `cluster.health`.

Amazon EKS garantisce che almeno due istanze del piano di controllo (control-plane) Kubernetes su tre siano nodi del cluster integri e operativi, al fine di mantenere la disponibilità del cluster locale e prevenire l'interruzione del servizio. Se un cluster locale è bloccato in uno stato `UPDATING`, di solito è perché c'è qualche problema di infrastruttura o di configurazione che impedisce di garantire la disponibilità minima delle due istanze nel caso in cui il processo continui. Pertanto, il processo di aggiornamento interrompe l'avanzamento per evitare l'interruzione del servizio del cluster locale.

È importante risolvere i problemi di un cluster locale bloccato nello stato `UPDATING` e risolvere la causa principale in modo che il processo di aggiornamento possa essere completato e ripristinare il cluster locale nello stato `ACTIVE` con l'elevata disponibilità di tre istanze del piano di controllo (control-plane) Kubernetes.

Non terminate alcuna `Kubernetes` istanza di cluster locale EKS gestita su Outposts a meno che non sia esplicitamente indicato dall'assistenza. AWS Ciò è particolarmente importante per i cluster locali bloccati nello `UPDATING` stato perché è molto probabile che un altro nodo del piano di controllo non sia completamente integro e la chiusura dell'istanza sbagliata potrebbe causare l'interruzione del servizio e la perdita di dati del cluster locale.

I problemi più comuni sono i seguenti:

- Una o più istanze del piano di controllo non sono in grado di connettersi a Systems Manager a causa di una modifica della configurazione di rete effettuata dopo la prima creazione del cluster locale. Puoi eseguire una verifica chiamando `aws ssm start-session --target instance-id` da un host bastione nella regione. Se il comando non funziona, controlla se Systems Manager è in esecuzione sull'istanza del piano di controllo (control-plane).
- Impossibile creare nuove istanze del piano di controllo (control-plane) a causa delle autorizzazioni chiave KMS per i volumi EBS. Con le chiavi KMS gestite dall'utente per i volumi EBS crittografati, le istanze del piano di controllo (control-plane) termineranno se la chiave non è accessibile. Se le istanze vengono terminate, passa a una chiave KMS AWS gestita o assicurati che la politica delle chiavi gestite dall'utente conceda le autorizzazioni necessarie per il ruolo del cluster.
- Le istanze del piano di controllo (control-plane) di Systems Manager potrebbero aver perso l'accesso a internet. Verifica se la sottorete fornita durante la creazione del cluster dispone di un gateway NAT e un VPC con un gateway Internet. Usa VPC Reachability Analyzer per verificare se l'istanza del control plane può raggiungere il gateway Internet. Per ulteriori informazioni, consulta la sezione Getting started with VPC Reachability Analyzer (Nozioni di base su VPC Reachability Analyzer). Se le reti private non dispongono di una connessione internet in uscita, assicurati che tutti gli endpoint VPC e gli endpoint gateway richiesti siano ancora presenti nella sottorete regionale del cluster (consulta the section called "Accesso tramite sottorete a AWS services").
- L'ARN del ruolo fornito è privo di alcune policy. Controlla se la policy AWS gestita: è AmazonEKSLocalOutpostClusterPolicy stata rimossa dal ruolo.
- Una delle nuove istanze del piano di controllo (control-plane) Kubernetes potrebbe aver subito un errore di bootstrap imprevisto. Invia un ticket al centro di Supporto AWS per ulteriori indicazioni sulla risoluzione dei problemi e sulla raccolta dei log in questo caso eccezionale.

Impossibile unire i nodi a un cluster

- Problemi relativi alle AMI:
  - Stai utilizzando un'AMI incompatibile. Sono supportate solo le AMI Amazon Linux 2023 ottimizzate per Amazon EKS. Per ulteriori informazioni, consulta Creare nodi Amazon Linux su AWS Outposts.
  - Se hai utilizzato un AWS CloudFormation modello per creare i tuoi nodi, assicurati che non utilizzasse un'AMI non supportata.
- Manca l'autenticatore AWS IAM`ConfigMap`: se manca, devi crearlo. Per ulteriori informazioni, consulta the section called "Applica aws-auth al tuo cluster `ConfigMap`".

- È stato utilizzato un gruppo di sicurezza non corretto: assicurati di utilizzare `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`* per il gruppo di sicurezza dei nodi worker. Il gruppo di sicurezza selezionato viene modificato AWS CloudFormation per consentire un nuovo gruppo di sicurezza ogni volta che viene utilizzato lo stack.
- A seguito dei passaggi del VPC di collegamento privato imprevisto: vengono passati dati CA (`--b64-cluster-ca`) o API Endpoint (`--apiserver-endpoint`) errati.

Raccolta di registri

Quando un Outpost viene disconnesso dalla AWS regione a cui è associato, è probabile che il cluster Kubernetes continui a funzionare normalmente. Tuttavia, se il cluster non funziona correttamente, segui i passaggi per la risoluzione dei problemi in Preparare i cluster Amazon EKS locali su Outposts per le disconnessioni di rete. AWS Se riscontri altri problemi, contatta l'assistenza. AWS AWS L'assistenza può guidarti nel download e nell'esecuzione di uno strumento di raccolta dei log. In questo modo, puoi raccogliere i log dalle istanze del piano di controllo del cluster Kubernetes e inviarli al AWS supporto per ulteriori indagini.

Le istanze del piano di controllo non sono raggiungibili tramite AWS Systems Manager

Quando le istanze del piano di controllo di Amazon EKS non sono raggiungibili tramite AWS Systems Manager (Systems Manager), Amazon EKS visualizza il seguente errore per il cluster.

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

Per risolvere questo problema, assicurati che il tuo VPC e le sottoreti soddisfino i requisiti in Creazione di un VPC e sottoreti per i cluster Amazon EKS su AWS Outposts e di aver completato i passaggi descritti nella Configurazione di Session Manager nella Guida per l'utente di Systems Manager. AWS

# Panoramica dei cluster Amazon EKS locali su AWS Outposts configurati con l'archivio di istanze EC2

Puoi utilizzare i cluster locali per eseguire l'intero cluster Amazon EKS localmente su Outposts. AWS Questo aiuta a mitigare il rischio di inattività delle applicazioni che potrebbero derivare da disconnessioni temporanee della rete al cloud e può aiutare a soddisfare i requisiti di residenza

dei dati. Poiché l'intero cluster Kubernetes viene eseguito localmente su Outposts, le applicazioni rimangono disponibili ed è possibile eseguire operazioni sul cluster durante le disconnessioni di rete dal cloud.

I cluster locali sono generalmente disponibili per l'uso con i rack di Outposts.

## Cluster locali Amazon EKS con archivio di istanze EC2 Outposts

I cluster locali Amazon EKS supportano AWS Outposts configurati con lo store di istanze EC2. Quando esegui cluster locali su Outposts con l'archivio di istanze EC2, Amazon EKS implementa un'architettura di cluster locali aggiornata che offre una maggiore parità con i cluster cloud Amazon EKS.

L'architettura aggiornata utilizzata con Outposts con supporto da store di istanze EC2 aggiunge quanto segue:

- Piano di controllo gestito. Il piano di controllo di Kubernetes viene eseguito in un account di servizio AWS gestito sul tuo Outpost. Amazon EKS gestisce il ciclo di vita del piano di controllo, inclusi gli aggiornamenti e i backup delle versioni della piattaforma Amazon EKS. `etcd`
- Maggiore parità di funzionalità con Amazon EKS nel cloud. Usa i componenti aggiuntivi di Amazon EKS, i ruoli IAM per gli account di servizio, l'identità del pod EKS, l'autenticazione ??? OIDC e. the section called “Voci di accesso” Usa Bottlerocket come sistema operativo del nodo di lavoro, oltre ai nodi di lavoro basati su Amazon Linux 2023.
- Allineato al ciclo di vita della versione di Amazon EKS Kubernetes. Le nuove versioni di Kubernetes e le versioni della piattaforma Amazon EKS diventano disponibili per i cluster locali con la stessa cadenza di Amazon EKS nel cloud. Supporto esteso Le versioni Kubernetes sono disponibili con gli stessi prezzi dei cluster cloud.
- Disponibile in tutte le AWS regioni commerciali che supportano i rack Outposts. AWS

> **Note**
>
> Se il tuo Outpost è configurato con Amazon EBS anziché con lo store di istanze EC2, l'architettura descritta in questo argomento non è disponibile per il tuo Outpost. Outposts configurato con EBS continuerà a utilizzare l'implementazione dei cluster locali esistenti. Per ulteriori informazioni, consulta the section called “Esecuzione di cluster locali”.
> Se sei interessato a creare un cluster locale su un EBS-backed Outpost utilizzando l'architettura dei cluster locali aggiornata, contatta il team del tuo account. AWS

Per un confronto funzionalità per funzionalità tra cluster locali e cluster estesi, consulta. the section called "Confronto tra le opzioni di implementazione"

## Nozioni di base

Per creare un cluster locale su un Outpost configurato con l'archivio di istanze EC2, consulta. the section called "Implementa un cluster locale (store di istanze EC2)"

Prima di creare un cluster, controlla:

- the section called "Crea un VPC e delle sottoreti (store di istanze EC2)"— Configurazione di VPC e sottorete per i cluster locali.
- the section called "Considerazioni sulla capacità (EC2 instance store)"— Selezione e posizionamento del piano di controllo e del tipo di `etcd` istanza.
- the section called "Preparati alle disconnessioni (archivio di istanze EC2)"— Autenticazione, osservabilità e considerazioni operative durante le disconnessioni.
- the section called "Componenti aggiuntivi di Amazon EKS (archivio di istanze EC2)"— Il set convalidato di componenti aggiuntivi Amazon EKS per cluster locali.

## Implementa un cluster locale Amazon EKS su AWS Outposts configurati con l'instance store EC2

Questo argomento fornisce istruzioni per la distribuzione di un cluster Amazon EKS locale su un AWS Outpost configurato con EC2 instance store.

> **Note**
>
> Se Outpost è configurato con Amazon EBS anziché con EC2 instance store, l'architettura descritta in questo argomento non è disponibile per Outpost. Outposts configurati con EBS continueranno a utilizzare l'implementazione dei cluster locali esistente. Per ulteriori informazioni, consulta the section called "Implementazione di un cluster locale".
> Se sei interessato a creare un cluster locale su EBS-backed Outpost utilizzando l'architettura dei cluster locali aggiornata, contatta il team del tuo account. AWS

**Important**

- I cluster locali supportano solo i rack dell'Outpost. Un singolo cluster locale può essere eseguito su più dell'Outpost fisici che comprendono un unico Outpost logico. Un singolo cluster locale non può essere eseguito su più Outpost logici.
- Il piano di controllo Kubernetes funziona in un account di servizio AWS gestito su Outpost. Non puoi accedere alle istanze del piano di controllo, visualizzarle nella console Amazon EC2 o eseguire carichi di lavoro su di esse.
- Il piano di controllo funziona su 6 istanze EC2 (`etcd`3+3 server API) in una topologia etcd esterna. Per i requisiti di capacità, consulta. the section called "Considerazioni sulla capacità (EC2 instance store)"
- Il plug-in Amazon VPC CNI per Kubernetes, kube-proxy e CoredNS viene installato automaticamente nei cluster locali. Per l'elenco completo dei componenti aggiuntivi convalidatithe section called "Componenti aggiuntivi di Amazon EKS (archivio di istanze EC2)", consulta.
- Amazon EKS esegue periodicamente il backup`etcd`. In caso di `etcd` perdita del quorum, Amazon EKS ripristina lo stato del cluster dallo snapshot più recente.

Prerequisiti

- Un rack Outpost esistente configurato con EC2 instance store, con capacità virtualizzata sufficiente per le istanze del piano di controllo. Per informazioni, consulta the section called "Considerazioni sulla capacità (EC2 instance store)".
- Un VPC e sottoreti che soddisfano i requisiti descritti in. the section called "Crea un VPC e delle sottoreti (store di istanze EC2)"
- Lo strumento a riga di comando `kubectl` è installato sul computer o AWS CloudShell. La versione può essere la stessa o fino a una versione secondaria precedente o successiva alla versione Kubernetes del cluster. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- La versione AWS CLI `2.35.3` o successiva installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version`. Per installare o aggiornare, consulta Installazione della AWS CLI.

- Un principale IAM (utente o ruolo) con le autorizzazioni per creare e descrivere un cluster Amazon EKS. Per ulteriori informazioni, consultare the section called "Creazione di un cluster Kubernetes locale su un Outpost" e the section called "Elencare o descrivere tutti i cluster".
- Un ruolo IAM per il cluster con le seguenti politiche AWS gestite allegate: AmazonEKSClusterPolicye AmazonEKSServicePolicy. La policy di fiducia dei ruoli deve consentire al responsabile del `eks.amazonaws.com` servizio di assumere il ruolo.

> **Note**
>
> Quando crei un cluster locale, il principale IAM che crea il cluster viene aggiunto in modo permanente alla tabella di autorizzazione RBAC di Kubernetes come amministratore con autorizzazioni. `system:masters` Assicurati di non eliminare mai questo preside.

## Crea un cluster locale

Puoi creare un cluster locale utilizzando la AWS CLI Console di gestione AWS, l'API Amazon EKS o. AWS CloudFormation

Quando crei un cluster locale, tieni presente quanto segue:

- È necessario passare le sottoreti nella zona di disponibilità in cui si trova l'Outpost. La richiesta ha esito negativo se si passano sottoreti in più zone di disponibilità.
- È necessario impostare `endpointPrivateAccess` su `true`. Opzionalmente puoi anche abilitare. `endpointPublicAccess`
- È necessario passare valori per`outpostConfig`, tra cui `outpostArn`s`controlPlaneInstanceType`, e`etcdInstanceType`. Facoltativamente, è possibile specificare la configurazione di posizionamento. Per informazioni, consulta the section called "Considerazioni sulla capacità (EC2 instance store)".

AWS CLI

```
aws eks create-cluster \
  --name my-cluster \
  --role-arn arn:aws: iam::111122223333:role/myEKSClusterRole \
  --kubernetes-version 1.36 \
```

```
  --resources-vpc-config subnetIds=subnet-
ExampleID1,endpointPrivateAccess=true,endpointPublicAccess=true \
  --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}' \
  --access-config
 authenticationMode=API_AND_CONFIG_MAP,bootstrapClusterCreatorAdminPermissions=true \
  --outpost-config outpostArns=arn:aws: outposts:region-code:111122223333:outpost/op-
uniqueid,controlPlaneInstanceType=m5.large,controlPlanePlacement={spreadLevel=host},etcdInstanc
```

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Crea cluster.
3. Per la posizione del piano di controllo Kubernetes, scegli Outposts. AWS
4. Configura quanto segue:
   - Nome: un nome per il cluster.
   - Versione Kubernetes: la versione da usare.
   - Ruolo del servizio cluster: il ruolo IAM con e. AmazonEKSClusterPolicyAmazonEKSServicePolicy
   - Outpost ID: l'Outpost logico in cui implementare.
   - Tipo di istanza del piano di controllo: il tipo di istanza per il server, lo scheduler e il controller manager dell'API Kubernetes.
   - etcd instance type — Il tipo di istanza per. `etcd`
   - Livello di diffusione, `host` o `rack` per il piano di controllo e il `etcd` posizionamento.
5. Configurazione della rete: VPC, sottoreti, gruppi di sicurezza. Abilita l'accesso privato agli endpoint.
6. Configura l'osservabilità: registrazione del piano di controllo.
7. Revisione e creazione.

### API **`CreateCluster`**

```
{
    "name": "my-cluster",
    "roleArn": "arn:aws: iam::111122223333:role/myEKSClusterRole",
    "version": "1.36",
    "resourcesVpcConfig": {
```

```
        "subnetIds": ["subnet-ExampleID1"],
        "endpointPublicAccess": true,
        "endpointPrivateAccess": true
    },
    "logging": {
        "clusterLogging": [{
            "types": ["api", "audit", "authenticator", "controllerManager",
 "scheduler"],
            "enabled": true
        }]
    },
    "accessConfig": {
        "authenticationMode": "API_AND_CONFIG_MAP",
        "bootstrapClusterCreatorAdminPermissions": true
    },
    "outpostConfig": {
        "outpostArns": ["arn:aws: outposts:region-code:111122223333:outpost/op-
uniqueid"],
        "controlPlaneInstanceType": "m5.large",
        "controlPlanePlacement": {
            "spreadLevel": "host"
        },
        "etcdInstanceType": "m5.large",
        "etcdPlacement": {
            "spreadLevel": "host"
        }
    }
}
```

I `etcdInstanceType` parametri `controlPlaneInstanceType` and sono obbligatori quando si crea un cluster locale su Outposts configurato con EC2 instance store. I parametri `controlPlanePlacement` e `etcdPlacement` sono facoltativi. Se non li specifichi, non viene applicata alcuna strategia di distribuzione del posizionamento.

> **Note**
>
> `outpostConfig`è una proprietà di sola creazione. Per modificare il tipo di istanza del piano di controllo, il tipo di `etcd` istanza, il posizionamento o Outpost, è necessario creare un nuovo cluster.

## Connect al cluster

Configura `kubectl` per comunicare con il tuo cluster:

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

> **Note**
>
> Se hai creato il tuo cluster utilizzando le versioni AWS CLI seguenti`2.35.3`, devi aggiornare il tuo `kubeconfig` per specificarlo `--cluster-name` (anziché`--cluster-id`) negli args `exec`

L'endpoint del server API Kubernetes del cluster è ospitato in Amazon Route 53. L'endpoint si risolve negli indirizzi IP privati delle interfacce di rete elastiche (ENI) tra account che Amazon EKS crea nelle tue sottoreti. Questi ENI hanno indirizzi IP privati fissi che non cambiano durante il ciclo di vita del cluster.

Quando crei un cluster, Amazon EKS crea 3 interfacce di rete elastiche nelle sottoreti specificate. Queste interfacce di rete consentono la comunicazione tra il cluster e il VPC. Ogni interfaccia di rete ha il testo `Amazon EKS` *`cluster-name`* nella sua descrizione.

## Distribuisci nodi di lavoro

Distribuisci i nodi di lavoro nel tuo cluster locale utilizzando gruppi di nodi autogestiti.

1. Implementa il AWS CloudFormation modello di nodo di lavoro o crea istanze EC2 individuali sul tuo Outpost.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-24/
   amazon-eks-outpost-nodegroup.yaml
   ```

2. Concedi all'istanza del nodo l'accesso al ruolo del cluster. Crea una voce di accesso per il ruolo IAM del nodo:

   ```
   aws eks create-access-entry \
     --cluster-name my-cluster \
     --principal-arn arn:aws: iam::111122223333:role/myNodeRole \
     --type EC2_LINUX
   ```

In alternativa, se utilizzi il comando `aws-auth` ConfigMap for authentication, aggiungi il ruolo dell'istanza del nodo a ConfigMap. Per informazioni, consulta the section called "Accesso API Kubernetes".

3. Verifica che i tuoi nodi siano nello `Ready` stato:

```
kubectl get nodes
```

### Risorse interne

Quando crei un cluster locale, Amazon EKS installa automaticamente le versioni autogestite dei seguenti componenti aggiuntivi nello spazio dei `kube-system` nomi:`coredns`, `kube-proxy` e (`aws-node`VPC CNI). Facoltativamente, puoi installare le versioni gestite di questi componenti aggiuntivi tramite l'API dei componenti aggiuntivi di Amazon EKS. Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi di Amazon EKS (archivio di istanze EC2)".

Amazon EKS crea anche interfacce di rete elastiche (ENI) tra account nelle sottoreti per la comunicazione cluster-VPC. Non eliminare queste interfacce di rete. Se viene eliminato un ENI su più account o il relativo indirizzo IP cambia, ogni nodo e ogni amministratore che utilizza l' IP-based accesso statico devono essere aggiornati manualmente.

## Crea un VPC e sottoreti per i cluster locali di Amazon EKS su AWS Outposts configurati con l'instance store EC2

Durante la creazione di un cluster locale, in genere si specificano un VPC e almeno una sottorete privata in esecuzione su Outposts. Questo argomento fornisce una panoramica dei requisiti del VPC e della sottorete per il cluster locale.

> **Note**
>
> Se il tuo Outpost è configurato con Amazon EBS anziché con EC2 instance store, l'architettura descritta in questo argomento non è disponibile per il tuo Outpost. Outposts configurati con EBS continueranno a utilizzare l'implementazione dei cluster locali esistente. Per ulteriori informazioni, consulta the section called "Creazione di un VPC e delle sottoreti". Se sei interessato a creare un cluster locale su EBS-backed Outpost utilizzando l'architettura dei cluster locali aggiornata, contatta il team del tuo account. AWS

## Requisiti VPC

- Il VPC deve disporre di indirizzi IP sufficienti per il cluster locale, tutti i nodi e le altre risorse Kubernetes che desideri creare. Puoi associare blocchi CIDR aggiuntivi al tuo VPC prima o dopo la creazione del cluster. Il riconoscimento di un nuovo blocco CIDR associato può richiedere fino a 1 ora.
- Il VPC deve disporre del nome host DNS e del supporto per la risoluzione DNS. In caso contrario, i nodi non possono registrarsi al cluster. Per ulteriori informazioni, consulta Attributi DNS per il VPC nella Guida per l'utente di Amazon VPC.
- Per accedere al cluster locale tramite la rete locale, il VPC deve essere associato alla tabella di instradamento del gateway locale dell'Outpost. Per ulteriori informazioni, consulta le associazioni VPC nella Guida per l'utente di  AWS Outposts.

## Requisiti della sottorete

Quando crei un cluster locale, specifichi almeno una sottorete privata su Outpost. Amazon EKS crea tre interfacce di rete elastiche tra account nelle sottoreti specificate. Queste interfacce di rete consentono la comunicazione tra il cluster e il VPC.

Le sottoreti specificate devono soddisfare i seguenti requisiti:

- Tutte le sottoreti devono trovarsi sullo stesso Outpost logico.
- Tutte le sottoreti devono trovarsi nello stesso VPC.
- Tutte le sottoreti devono trovarsi nella zona di disponibilità in cui è ospitato l'Outpost.
- Ogni sottorete deve avere almeno 3 indirizzi IP disponibili per le interfacce di rete elastiche multiaccount di Amazon EKS.
- Le sottoreti devono utilizzare la denominazione basata sull'indirizzo IP. La denominazione basata sulle risorse di Amazon EC2 non è supportata da Amazon EKS.
- Le sottoreti devono disporre di un percorso verso il gateway locale del rack Outpost per accedere al server dell'API Kubernetes tramite la rete locale. Se le sottoreti non dispongono di un percorso verso il gateway locale, devi comunicare con il server API Kubernetes dall'interno del VPC.
- Le sottoreti non possono risiedere nelle seguenti zone di disponibilità.

| AWS Regione | Nome della Regione | ID della zona di disponibilità non consentita |
|---|---|---|
| us-east-1 | Stati Uniti orientali (Virginia settentrionale) | use1-az3 |
| us-west-1 | Stati Uniti occidentali (California settentrionale) | usw1-az2 |
| ca-central-1 | Canada (Centrale) | cac1-az3 |

### Accesso alla sottorete a AWS services

I cluster locali necessitano di connettività alla AWS regione per le operazioni di gestione dei cluster, i `etcd` backup e gli aggiornamenti del piano di controllo. In uno stato disconnesso, il cluster locale può continuare a funzionare, ma le operazioni di gestione del cluster che Amazon EKS può eseguire sono limitate. Per ulteriori informazioni, consulta the section called "Preparati alle disconnessioni (archivio di istanze EC2)".

## Seleziona i tipi di istanza e il posizionamento per i cluster locali Amazon EKS su AWS Outposts configurati con l'instance store EC2

Questo argomento fornisce indicazioni per la selezione dei tipi di istanze del piano di controllo Kubernetes e la configurazione del posizionamento per soddisfare i requisiti di alta disponibilità per il cluster Amazon EKS locale su un AWS Outpost configurato con EC2 instance store.

> **Note**
>
> Se il tuo Outpost è configurato con Amazon EBS anziché con EC2 instance store, l'architettura descritta in questo argomento non è disponibile per il tuo Outpost. Outposts configurati con EBS continueranno a utilizzare l'implementazione dei cluster locali esistente. Per ulteriori informazioni, consulta the section called "Considerazioni sulla capacità".
> Se sei interessato a creare un cluster locale su EBS-backed Outpost utilizzando l'architettura dei cluster locali aggiornata, contatta il team del tuo account. AWS

## Architettura del piano di controllo

Il piano di controllo Kubernetes del cluster locale viene eseguito su 6 istanze EC2 su Outpost:

- 3 istanze del piano di controllo: ospitano i componenti del piano di controllo Kubernetes, tra cui il server API Kubernetes, lo scheduler e il controller manager.
- 3 **`etcd`** istanze: archivia lo stato del cluster Kubernetes in una topologia etcd esterna.

Queste istanze vengono eseguite in un account di servizio gestito su Outpost AWS. Non vengono visualizzate come istanze EC2 nel tuo AWS account o nella console Amazon EC2. La capacità consumata dal piano di controllo è visibile nella console AWS Outposts.

Il piano di controllo non si ridimensiona orizzontalmente sotto carico. Se è necessaria una dimensione diversa dell'istanza del piano di controllo, è necessario creare un nuovo cluster.

## Selezione del tipo di istanza

Con l'architettura aggiornata per i cluster locali Amazon EKS su AWS Outposts, il piano di controllo Kubernetes utilizza una topologia `etcd` dedicata: il server API ed è eseguito su istanze separate. `etcd` Quando crei un cluster, specifichi due tipi di istanza: `controlPlaneInstanceType` per il server API e per. `etcdInstanceType` `etcd`

I tipi di istanza scelti devono essere disponibili su Outpost. Le famiglie di istanze disponibili dipendono dalla generazione di Outpost e dallo SKU e famiglie di istanze diverse possono avere configurazioni di vCPU e memoria diverse a parità di dimensioni di istanza.

Server API**`controlPlaneInstanceType`** ()

| Nodi di lavoro previsti | vCPUs | Memoria |
|---|---|---|
| 1-20 | 2 | 8 GiB |
| 21-100 | 4 | 16 GiB |
| 101-250 | 8 | 32 GiB |
| 251-500 | 16 | 64 GiB |

**`etcd`** (**`etcdInstanceType`**)

| Nodi di lavoro previsti | vCPUs | Memoria |
|---|---|---|
| 1—50 | 2 | 8 GiB |
| 51—250 | 4 | 16 GiB |
| 251-500 | 8 | 32 GiB |

> **Important**
>
> Assicurati che Outpost abbia una capacità di 3 istanze `controlPlaneInstanceType` e 3 istanze `etcdInstanceType` per l'intera durata del cluster locale.

Utilizza queste tabelle come linea guida anziché come requisito rigoroso. Ti consigliamo di selezionare una famiglia di istanze e una dimensione che si adattino alla crescita prevista, poiché `controlPlaneInstanceType` non `etcdInstanceType` possono essere modificate dopo la creazione del cluster. Per modificare una delle due, devi creare un nuovo cluster.

## Piano di controllo e **`posizionamento ecc.`**

Per un'elevata disponibilità, consigliamo di distribuire il piano di controllo e `etcd` le istanze su più domini di errore hardware. È possibile controllare questa diffusione utilizzando la `spreadLevel` proprietà on `controlPlanePlacement` and `etcdPlacement` in. `outpostConfig` Una volta impostato`spreadLevel`, Amazon EKS crea un gruppo di collocamento EC2 con la strategia Spread. Non è necessario precreare un gruppo di collocamento.

> **Note**
>
> Il `groupName` parametro in non `controlPlanePlacement` viene utilizzato con Outposts che eseguono istanze EC2 basate su instance store. Se in precedenza utilizzavi un gruppo di collocamento con un cluster locale, ora lo usi invece. `spreadLevel`

Livelli di diffusione

La `spreadLevel` proprietà accetta due valori:

- **`host`**— Distribuisce le istanze del piano di controllo su diversi host fisici. Richiede almeno 3 host configurati con il tipo di istanza scelto.
- **`rack`**— Distribuisce le istanze del piano di controllo su diversi rack fisici Outpost. Richiede almeno 3 rack con host configurati con il tipo di istanza scelto.

### Pianificazione della capacità

Quando pianifichi la capacità di Outpost per i cluster locali, tieni conto di quanto segue:

- Piano di controllo e **`etcd`** istanze: 6 istanze per cluster (3 di `controlPlaneInstanceType` e 3 di). `etcdInstanceType`
- Nodi di lavoro: le istanze EC2 per i gruppi di nodi autogestiti.
- Host virtualizzati: il piano di controllo deve funzionare su host virtualizzati. Outpost deve disporre di host virtualizzati con una capacità sufficiente.

### Risoluzione dei problemi di posizionamento

Se il cluster rimane nello `UPDATING` stato `CREATING` o per un periodo prolungato dopo aver specificato un livello di spread, verifica che Outpost disponga di host o rack sufficienti con il tipo di istanza scelto per soddisfare la topologia di diffusione.

Per ulteriori informazioni sulla risoluzione dei problemi dei cluster locali, consulta. the section called "Risolvi i problemi relativi ai cluster (store di istanze EC2)"

## Prepara cluster Amazon EKS locali su AWS Outposts configurati con l'archivio di istanze EC2 per le disconnessioni di rete

Se il collegamento del servizio AWS Outposts che collega la rete locale al AWS cloud ha perso la connettività, puoi continuare a utilizzare il cluster Amazon EKS locale su un Outpost. Questo argomento spiega come preparare il cluster locale per le disconnessioni di rete e le relative considerazioni.

- I cluster locali consentono la stabilità e la continuità delle operazioni durante le disconnessioni di rete temporanee e non pianificate. AWS Outposts rimane un'offerta completamente connessa che funge da estensione del AWS cloud nel tuo data center. In caso di disconnessioni di rete tra Outpost e il AWS Cloud, ti consigliamo di provare a ripristinare la connessione. Per istruzioni,

consulta la lista di controllo per la risoluzione dei problemi di rete AWS nei rack di Outposts nella Guida per l'utente di Outposts. AWS

- Gli Outpost emettono un parametro `ConnectedStatus` che puoi utilizzare per monitorare lo stato di connettività del tuo Outpost. Per ulteriori informazioni, consulta Outposts Metrics nella Guida per l'utente di Outposts. AWS

## Autenticazione durante le disconnessioni di rete

I cluster locali supportano diversi meccanismi di autenticazione. La loro disponibilità durante le disconnessioni di rete varia:

| Meccanismo di autenticazione | Disponibile durante la disconnessione? |
|---|---|
| AWS IAM (voci di accesso, `aws-auth` ConfigMap) | No L'IAM richiede la connettività alla AWS Regione. |
| OIDC (fornitore fornito dal cliente) | Dipende dall'ubicazione del fornitore. Se il provider OIDC è raggiungibile dalla rete locale di Outpost, l'autenticazione continua a funzionare. |
| certificati client x.509 | Sì. I certificati vengono convalidati localmente dal server API Kubernetes. |
| IRSA (ruoli IAM per gli account di servizio) | No Vedere the section called "IRSA e Pod Identity durante le disconnessioni" |
| EKS Pod Identity | No Vedere the section called "IRSA e Pod Identity durante le disconnessioni" |

certificati client x.509

Per mantenere `kubectl` l'accesso durante le disconnessioni di rete, crea un certificato x.509 del client prima che avvenga la disconnessione.

Per creare un certificato di amministrazione:

1. Genera una chiave privata e una richiesta di firma del certificato (CSR):

```
openssl req -new -newkey rsa:4096 -nodes \
    -keyout admin.key -out admin.csr -subj "/CN=admin"
```

2. Crea una `CertificateSigningRequest` risorsa Kubernetes e approvala:

```
cat admin.csr | base64 | tr -d '\n' > admin.csr.b64
```

```
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
  request: <base64-encoded-csr>
  signerName: kubernetes.io/kube-apiserver-client
  usages:
    - client auth
```

```
kubectl apply -f admin-csr.yaml
kubectl certificate approve admin-csr
```

3. Recupera il certificato firmato:

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode >
 admin.crt
```

4. Crea un file `ClusterRoleBinding` per concedere l'accesso come amministratore:

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

5. Creane uno `kubeconfig` che utilizzi il certificato:

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server $APISERVER_ENDPOINT --embed-certs
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

### Risoluzione DNS degli endpoint del cluster durante le disconnessioni

L'endpoint del server API Kubernetes per un cluster locale è ospitato in Amazon Route 53 e si risolve negli indirizzi IP privati delle interfacce di rete elastiche (ENI) tra account che Amazon EKS crea nelle tue sottoreti. Questi ENI hanno indirizzi IP privati statici che non cambiano durante il normale funzionamento del cluster.

Durante una disconnessione dalla rete, Outpost non può raggiungere Route 53, quindi il nome host dell'endpoint del cluster non si risolve a meno che tu non abbia preparato un percorso di risoluzione locale. Tre categorie di client devono raggiungere il server API:

- Amministratori di cluster in esecuzione`kubectl`.
- Worker nodes (`kubelet`) che inviano i battiti cardiaci dei nodi e le specifiche di estrazione.
- **`kube-proxy`**su ogni nodo, che imposta gli IP di servizio del cluster.

#### Opzione 1: soluzione DNS locale (consigliata)

AWS consiglia di implementare una soluzione DNS locale che memorizzi nella cache i record degli endpoint del cluster e li serva mentre Outpost è disconnesso. È possibile eseguire il proprio server DNS nell'ambiente locale che memorizza nella cache i record degli endpoint del cluster.

Se utilizzi una soluzione DNS locale, ti consigliamo di indirizzare le tue AMI `kubeconfig` e quelle del tuo nodo di lavoro al nome host dell'endpoint del cluster (non agli indirizzi IP ENI) in modo che la risoluzione sia coerente con la soluzione DNS locale.

#### Opzione 2: accesso statico IP-based

Se non desideri eseguire una soluzione DNS locale, puoi utilizzare l' IP-based accesso statico.

- Amministratori: configurate il vostro account `kubeconfig` in modo che punti direttamente a un indirizzo IP privato ENI con più account. Trova gli ENI cercando le interfacce di rete con la descrizione `Amazon EKS` *`cluster-name`* nel tuo account. AWS L'indirizzo IP di ciascun ENI è stabile per tutta la durata del cluster in condizioni di normale funzionamento.
- Nodi di lavoro (AMI ottimizzate per Amazon EKS): quando avvii nodi di lavoro da un'AMI ottimizzata per Amazon EKS, lo script bootstrap aggiunge l'endpoint del cluster `/etc/hosts` agli indirizzi IP ENI. Non è necessaria alcuna configurazione aggiuntiva.
- Nodi di lavoro (AMI personalizzate): aggiungi il nome host dell'endpoint del cluster e gli indirizzi IP ENI `/etc/hosts` nel tuo bootstrap personalizzato. In caso contrario, `kubelet` non è `kube-proxy` possibile raggiungere il server API durante una disconnessione.

> **Important**
>
> Se un ENI con più account viene eliminato o il suo indirizzo IP cambia, ad esempio se lo elimini o lo modifichi in modo da impedire ad Amazon EKS di ricollegarlo, ogni nodo e ogni amministratore che utilizza IP-based l'accesso statico devono essere aggiornati manualmente. Con una soluzione DNS locale, non è richiesto alcun intervento manuale.

Risoluzione DNS del pod durante le disconnessioni

Per evitare errori DNS durante l'operazione di disconnessione, configura il modello di avvio del nodo di lavoro per sovrascrivere l'impostazione. `kubelet's `resolvConf` Nei tuoi dati utente, crea un `resolv.conf` file personalizzato (ad esempio`/etc/kubernetes/resolv.conf`) contenente solo `nameserver 10.0.0.2` (senza il dominio di ricerca VPC), quindi imposta il tuo. `spec.kubelet.config.resolvConf: /etc/kubernetes/resolv.conf` `NodeConfig` Ciò rimuove il dominio *`region-code`*`.compute.internal` di ricerca dalla configurazione DNS del pod, impedendo che le query vengano inoltrate all'irraggiungibile resolver DNS VPC quando sono disconnesse.

L'esempio seguente mostra i dati utente del nodo di lavoro:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
mkdir -p /etc/kubernetes
echo "nameserver 10.0.0.2" > /etc/kubernetes/resolv.conf

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    ...
```

```
  kubelet:
    config:
      resolvConf: /etc/kubernetes/resolv.conf

--BOUNDARY--
```

IRSA e Pod Identity durante le disconnessioni

> **Important**
>
> IRSA ed EKS Pod Identity dipendono da AWS STS, che funziona nella regione. AWS Durante una disconnessione dalla rete, i carichi di lavoro che utilizzano IRSA o Pod Identity non possono ottenere nuove credenziali. Le credenziali esistenti scadono dopo un certo periodo di tempo.

Non è consigliabile assumere dipendenze funzionali o operative dai Region-based AWS servizi per i carichi di lavoro che devono rimanere disponibili durante le disconnessioni di rete.

## `comportamento etcd durante le disconnessioni`

Durante le disconnessioni di rete, non è possibile eseguire il backup delle `etcd` istantanee. Se più di un'`etcd`istanza diventa non disponibile durante una disconnessione, `etcd` perde il quorum e le operazioni dell'API Kubernetes non sono disponibili finché Outpost non si riconnette e il quorum non viene ripristinato. `etcd` I carichi di lavoro già in esecuzione continuano a funzionare.

### Registrazione del piano di controllo durante le disconnessioni

Durante le disconnessioni di rete, i log del piano di controllo vengono memorizzati nella cache locale sulle istanze del piano di controllo. Quando la connettività viene ripristinata, i log vengono inviati ad Amazon CloudWatch Logs nella regione principale. AWS Non è necessario installare o mantenere alcun agente di registrazione sul piano di controllo.

### Osservabilità locale

Puoi monitorare il cluster localmente durante le disconnessioni utilizzando Prometheus, Grafana o altre soluzioni di terze parti per analizzare l'endpoint delle metriche del server API Kubernetes.

## Archivio di immagini locale

Per scalare le distribuzioni con repliche aggiuntive o per ripristinare gli errori del pod durante le disconnessioni, è necessario disporre di un repository di immagini del contenitore locale (ad esempio un registro Docker) oppure le immagini devono essere memorizzate nella cache sul nodo prima della disconnessione. Amazon ECR non è disponibile durante le disconnessioni di rete.

## Ottimizza il comportamento di failover dei pod Kubernetes

Durante una disconnessione dalla rete, il piano di controllo di Kubernetes non può comunicare con la regione. AWS Se un nodo diventa irraggiungibile, il comportamento predefinito di Kubernetes è quello di rimuovere i pod dopo un periodo di timeout. Puoi regolare questo comportamento utilizzando le tolleranze e le specifiche del tuo pod per controllare la velocità con cui i pod `tolerationSeconds` vengono riprogrammati durante le partizioni. Per indicazioni dettagliate ed esempi, consulta https://docs.aws.amazon.com/eks/latest/best-practices/hybrid-nodes-network-disconnection-best-practices.html # tune_kubernetes_pod_failover_behavior [Tune Kubernetes pod failover behavior] nella _Amazon EKS Best Practices Guide.

## Simula una disconnessione di rete

Prima di avviare la produzione con il cluster locale, simula una disconnessione per verificare di poter accedere al cluster quando è disconnesso.

1. Applica le regole del firewall sui dispositivi di rete che collegano Outpost alla regione. AWS Questo disconnette il link del servizio dell'Outpost.
2. Verifica la connessione al tuo cluster locale utilizzando il certificato x.509 che hai creato:

```
kubectl --kubeconfig admin.kubeconfig get nodes
```

> **Note**
>
> Se disponi di servizi già in produzione sul tuo Outpost, non simulare una disconnessione. La disconnessione del service link ha effetto su tutti i servizi in esecuzione su Outpost.

# Componenti aggiuntivi Amazon EKS per cluster locali su AWS Outposts configurati con l'archivio di istanze EC2

I cluster locali su AWS Outposts configurati con l'instance store EC2 supportano i componenti aggiuntivi di Amazon EKS. I seguenti componenti aggiuntivi sono stati convalidati per l'uso con cluster locali. I componenti aggiuntivi verranno convalidati nel tempo.

> **Note**
>
> Se il tuo Outpost è configurato con Amazon EBS anziché con lo store di istanze EC2, l'architettura descritta in questo argomento non è disponibile per il tuo Outpost. Outposts configurato con EBS continuerà a utilizzare l'implementazione dei cluster locali esistente, che non supporta i componenti aggiuntivi di Amazon EKS.
> Se sei interessato a creare un cluster locale su un EBS-backed Outpost utilizzando l'architettura dei cluster locali aggiornata, contatta il team del tuo account. AWS

## Componenti aggiuntivi convalidati

| Add-on | Description |
| --- | --- |
| Plug-in della CNI di Amazon VPC per Kubernetes | Fornisce una rete VPC nativa per i pod. |
| kube-proxy | Mantiene le regole di rete sui nodi per le comunicazioni con Kubernetes Service. |
| CoreDNS | Fornisce la risoluzione DNS per Kubernetes Services. |
| EKS Pod Identity Agent | Consente ai pod di utilizzare EKS Pod Identity per autenticarsi ai AWS servizi. |

Customer-managed le versioni del plug-in Amazon VPC`kube-proxy`, CNI e CoreDNS vengono installate automaticamente quando si crea un cluster locale. Facoltativamente, puoi installare le versioni gestite di questi componenti aggiuntivi.

> **Important**
>
> EKS Pod Identity e IRSA dipendono da AWS STS nella regione. AWS Durante le disconnessioni di rete, i carichi di lavoro che utilizzano questi meccanismi non possono ottenere nuove credenziali. Per ulteriori informazioni, consulta the section called "Preparati alle disconnessioni (archivio di istanze EC2)".

### Add-ons non ancora convalidato

Add-ons non elencati sopra non sono stati convalidati per l'uso con cluster locali. L'uso di componenti aggiuntivi non convalidati non è consigliato.

Man mano che i componenti aggiuntivi verranno convalidati, questa pagina verrà aggiornata.

### Plugin CNI alternativi

Amazon VPC CNI è il CNI ufficialmente supportato per le distribuzioni di Amazon EKS su Outposts. Se desideri utilizzare un plug-in CNI alternativo, consulta. the section called "Plug-in CNI alternativi"

## Risolvi i problemi relativi ai cluster Amazon EKS locali su AWS Outposts configurati con l'instance store EC2

Questo argomento descrive gli errori più comuni che potresti riscontrare durante l'utilizzo di cluster Amazon EKS locali su AWS Outposts configurati con EC2 instance store e come risolverli.

#### `kubectl non funziona dopo aver configurato kubeconfig`

Se vedi il seguente errore dopo l'esecuzione: `aws eks update-kubeconfig`

```
Either parameter --cluster-name or --cluster-id must be specified.
```

O:

```
couldn't get current server API group list: getting credentials: exec: executable aws
 failed with exit code 1
```

Verifica che your `kubeconfig` (`~/.kube/config`) specifichi `--cluster-name` negli `exec` args, non: `--cluster-id`

```
    args:
      - --region
      - region-code
      - eks
      - get-token
      - --cluster-name
      - my-cluster
      - --output
      - json
    command: aws
```

Nodi che non si uniscono al cluster

Se i nodi non vengono visualizzati durante l'esecuzione`kubectl get nodes`:

1. Controlla **`kubelet`** i log sul nodo. Accedi al nodo con SSM o SSH ed esegui:

   ```
   systemctl status kubelet -l
   ```

2. Verifica la presenza di errori non autorizzati. Gli errori non autorizzati possono indicare la mancanza di autorizzazioni o problemi relativi al ruolo del nodo. `aws-auth` ConfigMap Verifica che il ruolo dell'istanza del nodo sia mappato correttamente in. `aws-auth` ConfigMap
3. Verifica lo stato della CSR. Verifica che la richiesta di firma del certificato del nodo sia stata approvata:

   ```
   kubectl get csr
   ```

   Se la CSR è attiva`Pending`, approvala:

   ```
   kubectl certificate approve node-csr-EXAMPLE
   ```

### Cluster bloccato nello stato CREATING

Se il cluster entra nello stato FAILED:

- Capacità insufficiente: verifica che Outpost disponga di una capacità virtualizzata sufficiente per 6 istanze del piano di controllo (`etcd`3+ 3 server API) dei tipi di istanze specificati. I tipi di istanza devono essere inseriti nel tuo Outpost.

- La topologia di diffusione non può essere soddisfatta: se hai specificato una `spreadLevel` di`host`, verifica che almeno 3 host siano configurati con il tipo di istanza scelto. Se lo hai specificato`rack`, verifica che almeno 3 rack abbiano host con il tipo di istanza scelto.
- Problemi relativi alle sottoreti: verifica che le sottoreti specificate si trovino nella zona di disponibilità in cui è ospitato l'Outpost e che ogni sottorete abbia almeno 3 indirizzi IP disponibili. Per informazioni, consulta the section called “Crea un VPC e delle sottoreti (store di istanze EC2)”.

#### `Cluster bloccato in stato di AGGIORNAMENTO`

Se il cluster rimane nello `UPDATING` stato durante una versione di Kubernetes o un aggiornamento della versione della piattaforma:

- Capacità insufficiente: durante gli aggiornamenti, le istanze del piano di controllo vengono eliminate prima della reistanziazione. Se la capacità liberata viene consumata da un altro carico di lavoro prima di poter effettuare il provisioning dell'istanza sostitutiva, l'aggiornamento si blocca. Verifica che Outpost abbia una capacità disponibile per i tipi di istanza del Control Plane.

#### `ecc. perdita del quorum`

Se perdi la capacità di eseguire le operazioni dell'API Kubernetes e Outpost viene disconnesso:

- `etcd`potrebbe aver perso il quorum. Ciò si verifica se più di un'`etcd`istanza non è disponibile durante una disconnessione.
- I carichi di lavoro già in esecuzione continuano a funzionare, ma le operazioni dell'API Kubernetes non sono disponibili.
- Una volta ripristinata la connettività, Amazon EKS ripristinerà lo stato del cluster dallo `etcd` snapshot più recente.

Contatta il AWS Support Center.

## Crea nodi Amazon Linux su AWS Outposts

> **Important**
>
> Amazon EKS Local Clusters on Outposts supporta solo nodi creati dalle seguenti AMI Amazon EKS-optimized Amazon Linux 2023:

- Amazon Linux 2023 standard (`amazon-linux-2023/x86_64/standard`)
- Nvidia Amazon Linux 2023 accelerato () `amazon-linux-2023/x86_64/nvidia`
- Neurone accelerato Amazon Linux 2023 () `amazon-linux-2023/x86_64/neuron`
- Bottlerocket (solo per Outposts supportati da EC2 instance store)

AWS ha terminato il supporto per EKS AL2-optimized e AL2-accelerated AMI, a partire dal 26 novembre 2025. Sebbene sia comunque possibile continuare a utilizzare le AMI EKS AL2 dopo la data di fine del supporto (26 novembre 2025), dopo tale data EKS non rilascerà più nuove versioni o aggiornamenti di Kubernetes alle AMI AL2, comprese versioni secondarie, patch e correzioni di bug. Vedi questo per ulteriori informazioni sulla deprecazione di AL2.

In questo argomento viene descritto come avviare gruppi con scalabilità automatica di nodi Amazon Linux su un Outpost che si registrano con il cluster Amazon EKS. Il cluster può essere sul AWS Cloud o su un Outpost.

- Un Outpost esistente. Per ulteriori informazioni, consulta What is AWS Outposts.
- Un cluster Amazon EKS esistente. Per distribuire un cluster sul AWS cloud, consulta. the section called "Creazione di un cluster " Per implementare un cluster su un Outpost, consulta la sezione the section called "Esecuzione di cluster locali".
- Supponiamo di creare i nodi in un cluster sul AWS Cloud e di disporre di sottoreti nella AWS regione in cui sono abilitati Outposts AWS , Wavelength AWS o Local Zones. AWS Quindi, le sottoreti non devono essere state passate al momento della creazione del cluster. Se stai creando i nodi in un cluster su un Outpost, durante la creazione del cluster devi avere passato una sottorete Outpost.
- (Consigliato per i cluster sul AWS cloud) Il plug-in CNI Amazon VPC per Kubernetes è configurato con il proprio ruolo IAM a cui è associata la politica IAM necessaria. Per ulteriori informazioni, consulta the section called "Configurare per IRSA". I cluster locali non supportano i ruoli IAM per gli account di servizio.

Puoi creare un gruppo di nodi Amazon Linux autogestito con `eksctl` o Console di gestione AWS (con un AWS CloudFormation modello). Puoi anche utilizzare Terraform.

Puoi creare un gruppo di nodi autogestito per un cluster locale con i seguenti strumenti descritti in questa pagina:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

> **Important**
>
> - Self-managed il gruppo di nodi include istanze Amazon EC2 nel tuo account. Queste istanze non vengono aggiornate in automatico quando l'utente o Amazon EKS aggiorna la versione del piano di controllo (control-plane) per tuo conto. Nella console, un gruppo di nodi autogestito non riceve alcuna indicazione di aggiornamento. È possibile visualizzare la versione di `kubelet` installata su un nodo selezionando il nodo nella finestra di dialogo Nodi nella scheda Panoramica del cluster per determinare quali nodi devono essere aggiornati. È necessario aggiornare manualmente i nodi. Per ulteriori informazioni, consulta the section called "Metodi di aggiornamento".
> - I certificati utilizzati da kubelet sui nodi autogestiti vengono emessi con una scadenza di un anno. Per impostazione predefinita, la rotazione dei certificati non è abilitata (vedi: https://kubernetes.io/docs/reference/config-api/kubelet-config.v1beta1/ #kubelet -config-k8s-io-v1beta1-KubeletConfiguration), ciò significa che se hai un nodo autogestito in esecuzione per più di un anno, non sarà più in grado di autenticarsi sull'API Kubernetes.
> - Come best practice, consigliamo ai clienti di aggiornare regolarmente i propri gruppi di nodi autogestiti per ricevere vulnerabilità ed esposizioni comuni (CVE) e patch di sicurezza dalle più recenti AMI ottimizzate per Amazon EKS. L'aggiornamento delle AMI utilizzate nei gruppi di nodi autogestiti attiva anche la ricreazione dei nodi e assicura che non si verifichino problemi dovuti a certificati kubelet scaduti.
> - In alternativa, puoi anche abilitare la rotazione dei certificati client (vedi:https://kubernetes.io/docs/tasks/tls/certificate-rotation/) durante la creazione dei gruppi di nodi autogestiti per assicurarti che i certificati kubelet vengano rinnovati man mano che il certificato corrente si avvicina alla scadenza.

## `eksctl`

Per avviare nodi Linux autogestiti utilizzando **`eksctl`**

1. Installa la versione `0.215.0` o successiva dello strumento da riga di `eksctl` comando installato sul tuo dispositivo o. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Se il tuo cluster è sul AWS cloud e la policy IAM gestita da Amazoneks_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, ti consigliamo di assegnarlo a un ruolo IAM da associare invece all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section called "Configurare per IRSA". Se il cluster si trova sul tuo Outpost, la policy deve essere associata al tuo ruolo di nodo.
3. Il comando seguente crea un gruppo di nodi in un cluster esistente. Il cluster deve essere stato creato utilizzando `eksctl`. Sostituisci *`al-nodes`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o una cifra, ma può anche includere trattini e caratteri di sottolineatura per i caratteri rimanenti. Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e dell' AWS account in cui stai creando il cluster. Se il cluster si trova su un Outpost, sostituisci *`id`* con l'ID di una sottorete dell'Outpost. Se il cluster esiste nel AWS cloud, sostituiscilo *`id`* con l'ID di una sottorete che non hai specificato al momento della creazione del cluster. Sostituisci i valori di esempio con i tuoi valori. Per impostazione predefinita, i nodi vengono creati con la stessa versione Kubernetes del piano di controllo.

   Sostituisci *`instance-type`* con un tipo di istanza disponibile sul tuo Outpost.

   Sostituisci *`my-key`* con il nome della coppia di chiavi Amazon EC2 o della chiave pubblica. Questa chiave viene utilizzata per eseguire il SSH nei nodi dopo il loro avvio. Se non disponi di una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta la sezione relativa alle coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2.

   Crea il tuo gruppo di nodi con il comando seguente.

   ```
   eksctl create nodegroup --cluster my-cluster --name al-nodes --node-type instance-
   type \
       --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false \
       --node-volume-type gp2 --subnet-ids subnet-id \
       --node-ami-family AmazonLinux2023
   ```

   Se il cluster è distribuito sul AWS cloud:

- Il gruppo di nodi implementato può assegnare gli indirizzi `IPv4` ai pod da un altro intervallo CIDR rispetto a quello dell'istanza. Per ulteriori informazioni, consulta the section called "Rete personalizzata".
- Il gruppo di nodi che implementi non richiede l'accesso a internet in uscita. Per ulteriori informazioni, consulta the section called "Cluster privati".

Per un elenco completo di tutte le impostazioni predefinite e le opzioni disponibili, consulta AWS Outposts Support nella documentazione di `eksctl`.

- Se i nodi non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" la sezione Risoluzione dei problemi con i cluster e i nodi Amazon EKS e Risoluzione dei problemi dei cluster Amazon EKS locali su Outposts. the section called "Impossibile unire i nodi a un cluster" AWS
- Di seguito viene riportato un output di esempio. Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.

## Console di gestione AWS

Fase 1: avvia i nodi Linux autogestiti utilizzando la Console di gestione AWS

1. Scarica l'ultima versione del modello. AWS CloudFormation

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-24/
amazon-eks-outpost-nodegroup.yaml
```

2. Apri la AWS CloudFormation console.
3. Scegli Crea stack e quindi seleziona Con nuove risorse (standard).
4. In Specifica modello, seleziona Carica un file di modello e Scegli file. Seleziona il file `amazon-eks-outpost-nodegroup.yaml` scaricato in un passaggio precedente, quindi seleziona Next (Successivo).
5. Nella pagina Specifica i dettagli dello stack, immetti i parametri seguenti e scegli Successivo:
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *`al-nodes`*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra

lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.

- ApiServerEndpoint: Inserisci l'endpoint del server API Kubernetes, visibile nella console EKS o tramite API. DescribeCluster
- ClusterName: Inserisci il nome del tuo cluster. Se questo nome non corrisponde al nome del cluster, i nodi non possono unirsi al cluster.
- ClusterId: Inserisci l'id assegnato al cluster dal servizio EKS. Visibile tramite DescribeCluster API. Se questo ID non corrisponde al tuo ID del cluster, i tuoi nodi non possono unirsi al cluster.
- CertificateAuthority: Inserisci la stringa codificata in base64 dell'autorità di certificazione Kubernetes. Visibile nella console EKS o tramite API. DescribeCluster
- ServiceCidr: Accedi al CIDR di Kubernetes Services. Visibile nella console EKS o tramite API. DescribeCluster
- ClusterControlPlaneSecurityGroup: scegli il SecurityGroupsvalore dall' AWS CloudFormation output che hai generato quando hai creato il tuo VPC.

  Nella procedura seguente viene illustrata un'operazione per recuperare il gruppo applicabile.

  i. Aprire la Console Amazon EKS.

  ii. Scegli il nome del cluster.

  iii. Scegli la scheda Reti.

  iv. Utilizza il valore dei gruppi di sicurezza aggiuntivi come riferimento quando selezioni dall'ClusterControlPlaneSecurityGroupelenco a discesa.
- NodeGroupName: inserisci un nome per il tuo gruppo di nodi. Questo nome può essere utilizzato in seguito per identificare il gruppo di nodi con dimensionamento automatico creato per i tuoi nodi.
- NodeAutoScalingGroupMinSize: Inserisci il numero minimo di nodi su cui il gruppo Auto Scaling del nodo può scalare.
- NodeAutoScalingGroupDesiredCapacity: Inserisci il numero di nodi desiderato su cui scalare quando viene creato lo stack.
- NodeAutoScalingGroupMaxSize: Inserisci il numero massimo di nodi su cui il gruppo Auto Scaling del nodo può scalare orizzontalmente.
- NodeInstanceType: scegli un tipo di istanza per i tuoi nodi. Se il cluster è in esecuzione sul AWS cloud, per ulteriori informazioni, consultathe section called "Tipi di istanza Amazon EC2". Se il

cluster è in esecuzione su un Outpost, puoi selezionare solo un tipo di istanza disponibile sul tuo Outpost.

- NodeImageIdSSMParam: Pre-populated con il parametro Amazon EC2 Systems Manager di una recente AMI ottimizzata per Amazon EKS per una versione variabile di Kubernetes. Se si desidera utilizzare una diversa versione secondaria di Kubernetes supportata da Amazon EKS, è possibile sostituire *1.XX* con una versione supportata differente. Si consiglia di specificare la stessa versione Kubernetes del cluster.

  Per utilizzare un'AMI accelerata ottimizzata per Amazon EKS, aggiorna il *NodeImageIdSSMParam* valore al parametro SSM desiderato. Scopri come recuperare gli ID AMI EKS da SSM qui.

  > **Note**
  >
  > L'AMI del nodo Amazon EKS è basata su Amazon Linux. Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux in Amazon Linux security center scegliendo la scheda per la versione desiderata. Puoi anche abbonarti al feed RSS applicabile. Gli eventi di sicurezza e privacy includono una panoramica del problema, quali sono i pacchetti interessati e come aggiornare le istanze per risolvere il problema.

- NodeImageId: (Facoltativo) Se utilizzi un'AMI personalizzata (anziché un'AMI ottimizzata per Amazon EKS), inserisci un ID AMI del nodo per la tua AWS regione. Se specifichi un valore qui, questo sostituisce tutti i valori nel NodeImageIdSSMParamcampo.
- NodeVolumeSize: Specificate la dimensione del volume root per i nodi, in GiB.
- NodeVolumeType: Specificate un tipo di volume root per i nodi.
- KeyName: inserisci il nome di una coppia di chiavi SSH Amazon EC2 che puoi usare per connetterti tramite SSH ai tuoi nodi dopo il loro avvio. Se non hai già una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta la sezione relativa alle coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2.

  > **Note**
  >
  > Se non fornisci una key pair qui, la creazione dello AWS CloudFormation stack fallisce.

- DisableIMDSv1: ogni nodo supporta Instance Metadat Service versione 1 (IMDSv1) e IMDSv2 per impostazione predefinita. Puoi disabilitare IMDSv1. Per evitare che in futuro i nodi e i pod nel gruppo di nodi utilizzino IMDSv1, imposta DisableIMDSv1 su vero. Per ulteriori informazioni

su IMDS, consulta Configurazione del servizio di metadati dell'istanza. Per ulteriori informazioni sulle relative limitazioni dell'accesso ai nodi, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

- VpcId: inserisci l'ID per il VPC che hai creato. Prima di scegliere un VPC, consulta VPC requirements and considerations.
- Subnets: (Sottoreti) se il cluster è su un Outpost, scegli almeno una sottorete privata nel VPC. Prima di scegliere le sottoreti, consulta Considerazioni e requisiti relativi alla sottorete. È possibile visualizzare le sottoreti private aprendo ogni collegamento relativo alla sottorete dalla scheda Reti del cluster.

6. Seleziona le opzioni desiderate nella pagina Configura opzioni dello stack, quindi scegli Next (Avanti).
7. Seleziona la casella di controllo a sinistra di Riconosco che AWS CloudFormation potrebbe creare risorse IAM. , quindi scegli Create stack.
8. Al termine della creazione dello stack, selezionalo nella console e scegli Output.
9. Registra il NodeInstanceRoleper il gruppo di nodi che è stato creato. Ciò sarà utile quando configurerai i nodi di Amazon EKS.

Fase 2: abilitazione dell'aggiunta di nodi al cluster

1. Verifica se disponi già di una `ConfigMap` per `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrata una `ConfigMap` per `aws-auth`, aggiornala se necessario.
   a. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Aggiungi una nuova voce `mapRoles`, se necessario. Impostate il `rolearn` NodeInstanceRolevalore sul valore registrato nella procedura precedente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
```

```
        - system:bootstrappers
        - system:nodes
[...]
```

c. Salva il file ed esci dall'editor di testo.

3. Se hai ricevuto un messaggio di errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`", applica lo `ConfigMap` di stock.

   a. Scarica la mappa di configurazione.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. Nel `aws-auth-cm.yaml` file, impostate `rolearn` il NodeInstanceRolevalore sul valore registrato nella procedura precedente. Per eseguire questa operazione, utilizza un editor di testo o sostituisci *`my-node-instance-role`* tramite il comando seguente:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
   ```

   c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

4. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

Inserisci `Ctrl+C` per tornare a un prompt dello shell (interprete di comandi).

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso non autorizzato o negato (kubectl) " nell'argomento relativo alla risoluzione dei problemi.

Se i nodi non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" la sezione Risoluzione dei problemi con i cluster e i nodi Amazon EKS e Risoluzione dei

problemi dei cluster Amazon EKS locali su Outposts. the section called "Impossibile unire i nodi a un cluster" AWS

5. Installa il driver CSI per Amazon EBS. Per ulteriori informazioni, consulta Installazione su. GitHub Nella sezione Set up driver permission (Configura autorizzazione del driver), assicurati di seguire le istruzioni per l'opzione Using IAM instance profile (Utilizzo del profilo dell'istanza IAM). È necessario utilizzare la classe di archiviazione `gp2`. La classe di archiviazione `gp3` non è supportata.

   Per creare una classe di archiviazione `gp2` nel cluster, completa i seguenti passaggi.

   a. Per creare il file `gp2-storage-class.yaml`, emetti il seguente comando:

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

   b. Applica il file manifesto al cluster.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (Solo nodi GPU) Se hai scelto un tipo di istanza GPU e un'AMI accelerata ottimizzata per Amazon EKS, devi applicare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo *vX.X.X* con la versione desiderata prima di eseguire il comando seguente. NVIDIA/k8s-device-plugin

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Fase 3: Azioni aggiuntive

1. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.
2. Se il cluster è implementato su un Outpost, salta questo passaggio. Se il cluster è distribuito sul AWS cloud, le seguenti informazioni sono facoltative. Se la policy IAM gestita AmazonEKS_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, si consiglia, invece, di assegnarla a un ruolo IAM associato all'account di servizio Kubernetes `aws-node`. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

# Panoramica dell'intelligenza artificiale e del Machine Learning su Amazon EKS

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Amazon Elastic Kubernetes Service (Amazon EKS) è un servizio Kubernetes gestito che consente alle organizzazioni di implementare, gestire e scalare carichi di lavoro di intelligenza artificiale (AI) e machine learning (ML) con flessibilità e controllo senza precedenti. Poiché Amazon EKS è basato su Kubernetes upstream, puoi applicare le tue competenze Kubernetes esistenti integrandoti perfettamente con strumenti e servizi open source. AWS

Che tu stia addestrando modelli su larga scala, eseguendo inferenze online in tempo reale o distribuendo applicazioni di intelligenza artificiale generativa, Amazon EKS offre le prestazioni, la scalabilità e l'efficienza dei costi richieste dai tuoi progetti. AI/ML

## Perché usare Amazon EKS per AI/ML

Amazon EKS fornisce il controllo, le integrazioni, le prestazioni e la scalabilità necessarie per i AI/ML progetti. Basato su Kubernetes upstream e integrato con i servizi AWS , Amazon EKS ti aiuta a utilizzare le competenze Kubernetes esistenti orchestrando carichi di lavoro complessi. Per i team alle prime armi con le AI/ML implementazioni, le competenze Kubernetes esistenti vengono trasferite senza curve di apprendimento ripide.

Amazon EKS supporta qualsiasi cosa, dalla personalizzazione del sistema operativo alla scalabilità di calcolo, e promuove la flessibilità tecnologica che preserva la scelta per le decisioni future sull'infrastruttura. La piattaforma offre le prestazioni e le opzioni di ottimizzazione richieste dai AI/ML carichi di lavoro, incluse le seguenti funzionalità:

- Controllo completo del cluster: Fine-tune costi e configurazioni senza astrazioni nascoste.
- Sub-second latenza: esegui carichi di lavoro di inferenza in tempo reale in produzione.
- Personalizzazioni avanzate: configura GPU multiistanza, ottimizzazione della rete e ottimizzazione a livello di sistema operativo.
- Orchestrazione unificata: orchestrazione tra pipeline e ambienti locali, edge e cloud. AI/ML

- Ottimizzazione dei costi: utilizza la scalabilità automatica, la pianificazione nativa della GPU e diversi tipi di istanze di GPU e acceleratore.

## Casi d'uso principali

Amazon EKS supporta un'ampia gamma di AI/ML carichi di lavoro, inclusi i seguenti casi d'uso comuni:

- Inferenza: Self-host modelli su Amazon EKS per casi d'uso che richiedono tempi di risposta a bassa latenza.
- Inferenza in batch: elabora set di dati di grandi dimensioni in modo efficiente tramite processi pianificati.
- Addestramento dei modelli: Addestra modelli complessi su set di dati di grandi dimensioni per lunghi periodi di tempo.
- Ottimizzazione dei modelli: migliora i modelli open source con conoscenze proprietarie del settore.
- Pipeline di generazione aumentata di recupero (RAG): integra i processi di recupero e generazione.
- Agentic AI: distribuisci agenti con modelli ospitati su Amazon Bedrock, terze parti o su Amazon EKS.

## Casi di studio

I clienti scelgono Amazon EKS per vari motivi, come l'ottimizzazione dell'utilizzo della GPU o l'esecuzione di carichi di lavoro di inferenza con latenza inferiore al secondo, come dimostrato nei seguenti case study. Per un elenco di tutti i casi di studio relativi ad Amazon EKS, consulta AWS Customer Success Stories.

- Il BMW Group gestisce una delle flotte connesse più grandi al mondo, con oltre 25 milioni di veicoli connessi, ha costruito la sua piattaforma Connected AI su Amazon EKS con Ray per la formazione distribuita e Karpenter per la scalabilità automatica delle GPU, riducendo i tempi di formazione dei modelli da ore a 30 minuti a 5€ per sessione di formazione e supportando oltre 550 sviluppatori in oltre 60 casi d'uso dell'intelligenza artificiale.
- Booking.com, una delle piattaforme di viaggio più importanti al mondo, ha migrato il suo sistema di inferenza ML per il ranking di ricerca su Amazon EKS per sbloccare la scalabilità per la sperimentazione, elaborando fino a 250.000 richieste al secondo con una latenza di 40 ms p99,9.

- Unitary elabora 26 milioni di video ogni giorno utilizzando l'intelligenza artificiale per la moderazione dei contenuti. L'azienda richiede un throughput elevato e un'inferenza a bassa latenza e ha ottenuto una riduzione dell'80% dei tempi di avvio dei container, il che garantisce una risposta rapida agli eventi di scalabilità in base alle fluttuazioni del traffico.
- Synthesia offre la creazione di video con intelligenza artificiale generativa come servizio per consentire ai clienti di creare video realistici a partire da istruzioni di testo. L'azienda ha ottenuto un miglioramento di 30 volte nella produttività della formazione sui modelli ML.
- Ada Support, una società di automazione del servizio AI-powered clienti, ha ottenuto una riduzione del 15% dei costi di elaborazione insieme a un aumento del 30% dell'efficienza di elaborazione.
- Snorkel AI consente alle aziende di creare e adattare modelli di base e modelli linguistici di grandi dimensioni. L'azienda ha ottenuto risparmi sui costi di oltre il 40% implementando meccanismi di scalabilità intelligenti per le risorse GPU.
- Artera utilizza Amazon Elastic File System (Amazon EFS) e Amazon EKS per addestrare modelli di machine learning che personalizzano il trattamento del cancro utilizzando immagini bioptiche ad alta risoluzione.
- Anthropic utilizza la sua famiglia di modelli base Claude di punta su Amazon EKS e gestisce alcuni dei più grandi cluster EKS in produzione, costituiti da istanze AWS Trainium (trn2) e GPU NVIDIA per carichi di lavoro AI insieme ai processori Graviton per l'elaborazione di dati ad uso intensivo della CPU. AWS

## Struttura della guida

La guida include una serie di guide pratiche che puoi seguire passo dopo passo per distribuire e gestire i carichi di lavoro su Amazon EKS. AI/ML Ogni guida fornisce istruzioni e configurazioni che puoi implementare direttamente nel tuo ambiente.

Oltre alle istruzioni, la guida fornisce le informazioni di base e i concetti fondamentali necessari per ogni argomento. Include anche i collegamenti alla AWS documentazione e alle risorse pertinenti per i dettagli tecnici più approfonditi necessari.

## Inizia a usare AI/ML su Amazon EKS

Per iniziare a pianificare e utilizzare AI/ML piattaforme e carichi di lavoro su Amazon EKS, segui la the section called "Configura il cluster" sezione per creare un cluster Amazon EKS, inclusi i componenti Kubernetes richiesti, nel tuo account. AWS Una volta che l'ambiente è attivo e funzionante, puoi continuare con i passaggi successivi:

- the section called "Configura il cluster": crea il cluster Amazon EKS, il monitoraggio e l'infrastruttura bucket Amazon S3 da utilizzare in questa sezione.
- the section called "Inferenza": usa Amazon EKS per distribuire, configurare e iniziare a utilizzare un'applicazione di inferenza con un modello di linguaggio di grandi dimensioni (LLM).
- the section called "Configurazione del cluster": configura i cluster Amazon EKS ottimizzati per i AI/ML carichi di lavoro.
- the section called "Gestione della capacità": gestisci e ottimizza le risorse di calcolo per i carichi di lavoro di machine learning su Amazon EKS.
- the section called "Gestione dei dispositivi": Gestisci dispositivi hardware specializzati utilizzando Dynamic Resource Allocation (DRA) e plug-in per dispositivi.

# Configura il cluster Amazon EKS per i AI/ML carichi di lavoro

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS. AI/ML

Questa sezione ti guida nella creazione di un cluster Amazon EKS pronto per eseguire carichi di lavoro di inferenza, tra cui l'elaborazione con GPU, lo stack di monitoraggio e lo storage Amazon S3 per i pesi dei modelli, insieme alle autorizzazioni IAM necessarie. AWS

## Panoramica dell'architettura

La configurazione crea la seguente infrastruttura:

- Cluster EKS con GPU-enabled nodi: un cluster Karpenter-managed NodePool che fornisce dinamicamente le istanze G-family GPU utilizzando la capacità Spot con fallback. On-Demand
- Stack di monitoraggio: Prometheus analizza le metriche di cluster, nodi e GPU e le scrive in remoto su Amazon Managed Service for Prometheus (AMP). Grafana fornisce dashboard per la visualizzazione. NVIDIA DCGM Exporter aggiunge GPU-specific parametri tra cui utilizzo, memoria, temperatura, consumo energetico, larghezza di banda NVLink e attività tensoriale.
- Model weights S3 bucket: un bucket Amazon S3 per l'archiviazione dei pesi dei modelli, con un'associazione EKS Pod Identity che consente l'accesso ai pod di carico di lavoro. read/write

# Opzioni di elaborazione in cluster

La guida fornisce due percorsi per configurare il cluster. Scegline uno e seguilo costantemente in tutti i passaggi.

- EKS Auto Mode: un singolo comando esegue il provisioning di un cluster EKS con EKS Auto Mode abilitata. Tutti i componenti necessari vengono forniti immediatamente, tra cui il Karpenter-based ridimensionamento automatico, l'agente di monitoraggio dei nodi EKS, i pull rapidi dei container con SOCI e il plug-in per i dispositivi NVIDIA.
- Self-managed Karpenter — Ogni componente viene installato e configurato in modo esplicito: Karpenter tramite`eksctl`, riparazione automatica del nodo tramite il relativo feature gate, l'agente di monitoraggio dei nodi EKS come componente aggiuntivo EKS e il plug-in del dispositivo NVIDIA tramite Helm. È inoltre possibile creare una configurazione personalizzata `EC2NodeClass` che utilizza le AMI NVIDIA AL2023 e configura SOCI. EKS-optimized

## Cosa configurerai

| Fase | Description |
|---|---|
| Crea un cluster | Effettua il provisioning del piano di controllo EKS e dei componenti a livello di cluster necessari per i carichi di lavoro della GPU. |
| Crea nodi GPU con provisioning dinamico | Definisci una GPU dinamica che esegua il provisioning delle istanze G-family GPU NodePool man mano che i carichi di lavoro vengono pianifica ti. |
| Esegui il test con un pod di esempio | Convalida la configurazione end-to-end eseguendo un `nvidia-smi` pod che attivi Karpenter per eseguire il provisioning di un nodo. GPU-enabled |
| Aggiungi capacità riservata (opzionale) | Allega una prenotazione On-Demand di capacità (ODCR) al modulo riservato NodeClass per primo con riserva. Spot/On-Demand |
| Monitoraggio dell'inst allazione | Implementa kube-prometheus-stack (Prometheus + Grafana) con scrittura remota su AMP, oltre alle metriche NVIDIA DCGM Exporter for GPU. |

| Fase | Description |
| --- | --- |
| Crea un bucket di pesi del modello | Crea un bucket S3 e configura EKS Pod Identity in modo che i workload pod possano leggere e scrivere i pesi dei modelli. |

## Nozioni di base

Per istruzioni dettagliate sull'uso della CLI, consulta. AWS the section called "Command-line interfaccia (CLI)" Per istruzioni dettagliate sull'uso di Terraform, consulta. the section called "Terraform"

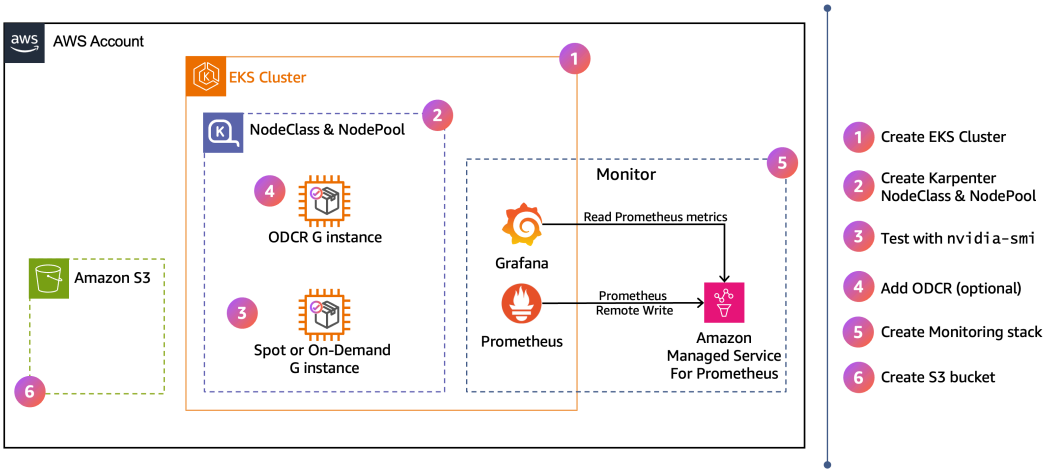
## Configura il cluster Amazon EKS per i AI/ML carichi di lavoro utilizzando le CLI

> **Tip**
>
> https://events.eksworkshop.com/workshops/genai/Registrati AI/ML per i prossimi workshop Amazon EKS.

Questa sezione illustra i passaggi per creare l'infrastruttura necessaria per eseguire carichi di lavoro di formazione o inferenza su Amazon EKS tramite comandi CLI. I passaggi includono la creazione di un cluster EKS, GPU-enabled nodi con EKS Auto Mode o Karpenter, uno stack di monitoraggio con Prometheus e Grafana e lo storage Amazon S3 per i pesi dei modelli.

Consulta la documentazione per EKS Auto Mode e Karpenter per ulteriori informazioni su come queste funzionalità forniscono e scalano automaticamente le istanze EC2 nei cluster EKS.

High-level architettura e flusso di lavoro




Il AWS diagramma mostra l'architettura di alto livello per la configurazione di questa sezione. I passaggi numerati sulla destra indicano l'ordine in cui si completa la configurazione nei passaggi seguenti.

## Prerequisiti

> **Important**
>
> Le risorse create in questo tutorial, tra cui cluster EKS, istanze GPU, Application Load Balancer e Amazon Managed Service for Prometheus, sono a pagamento. Elimina le risorse quando hai finito per evitare addebiti continui.

- `kubectl`>= 1,36. Per le istruzioni di configurazione, vedere. the section called "Configura kubectl ed eksctl"
- AWS CLI >= 2.27. Per le istruzioni di configurazione, vedere Installazione. https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
- Helm >= 3.14. Per le istruzioni di configurazione, vedere Setup Helm.
- `jq`. Per le istruzioni di configurazione, consulta Download jq.
- `eksctl`>= 0.227.0. Per le istruzioni di configurazione, vedere Installazione nella documentazione. `eksctl`

Verifica la tua `eksctl` versione:

```
eksctl version
```

Se utilizzi una versione precedente alla 0.227.0, segui la guida all'installazione di eksctl per eseguire l'aggiornamento alla versione più recente.

## Impostazione delle variabili di ambiente

Mantieni coerenti il nome e la AWS regione del cluster seguenti durante questi passaggi. La modifica può far sì che i comandi successivi indirizzino il cluster EKS sbagliato.

```
export CLUSTER_NAME=ai-eks-docs
export AWS_REGION=us-east-2
```

L'utilizzo di tutte le AZ disponibili migliora la tolleranza ai guasti e aumenta le possibilità di ottenere la capacità della GPU:

```
export AZS=$(aws ec2 describe-availability-zones \
  --region ${AWS_REGION} \
  --query "AvailabilityZones[?ZoneId!='use1-az3' && ZoneId!='usw1-az2' && ZoneId!
='cac1-az3'].ZoneName" \
  --output text | tr '\t' ',')
echo $AZS
```

> **Important**
>
> Le zone `use1-az3` di disponibilità e `cac1-az3` sono escluse perché Amazon EKS non supporta il posizionamento dei piani di controllo in tali zone. `usw1-az2` La creazione di un cluster con sottoreti in una di queste zone comporta un. `UnsupportedAvailabilityZoneException`

Output previsto:

```
us-east-2a,us-east-2b,us-east-2c
```

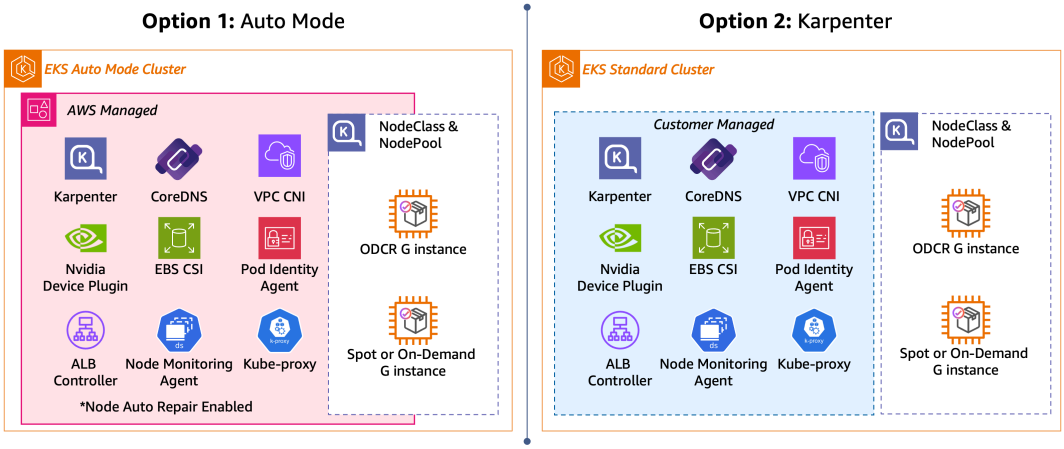
Gli AZ nell'output varieranno in base alla regione. Questo esempio mostra gli AZ disponibili per la `us-east-2` regione.

## Crea cluster e GPU NodePool

Questa sezione fornisce due percorsi per creare il cluster e i GPU-enabled nodi EKS, mostrati nel diagramma seguente. Scegli una sola opzione in tutta la guida.

- EKS Auto Mode: oltre ai principali componenti aggiuntivi di rete, storage e bilanciamento del carico, EKS Auto Mode include e gestisce le seguenti funzionalità per l'addestramento e l'inferenza dei carichi di lavoro: agente di monitoraggio dei nodi EKS, riparazione automatica dei nodi, snapshotter SOCI per una rapida estrazione dei container e predisposizione della GPU per l'impostazione predefinita. NodeClass Il plug-in del dispositivo NVIDIA è incluso nell'AMI accelerata Bottlerocket che EKS Auto Mode utilizza per i nodi. GPU-enabled
- Self-managed Karpenter — Su un cluster EKS senza EKS Auto Mode, sei responsabile dell'installazione e della configurazione dei componenti necessari per i carichi di lavoro di addestramento e inferenza. Ciò include componenti aggiuntivi di rete (VPC CNI, CoreDNS, kube-proxy), Karpenter, l'agente di monitoraggio dei nodi EKS, il plug-in per dispositivi NVIDIA e lo snapshotter SOCI per richiamare rapidamente i container.

Opzioni del cluster EKS: EKS Auto Mode e Karpenter autogestito




In ognuno dei passaggi seguenti, scegli un percorso (EKS Auto Mode, Karpenter) e seguilo per tutto il tempo. Dopo aver completato i passaggi per il percorso scelto, avrai un cluster EKS con una GPU NodePool pronto per pianificare i carichi di lavoro della GPU.

## Fase 1: Creazione del cluster

Inizia creando il tuo cluster EKS e installando i componenti del cluster necessari per i carichi di lavoro della GPU.

Con EKS Auto Mode, un singolo `eksctl create cluster --enable-auto-mode` comando esegue il provisioning di un cluster EKS pronto per i carichi di lavoro della GPU.

Con Karpenter autogestito, il `eksctl create cluster` comando fornisce i componenti aggiuntivi di rete principali, quindi sono necessari passaggi aggiuntivi per abilitare la riparazione automatica dei

nodi tramite un feature gate di Karpenter, installare l'agente di monitoraggio dei nodi EKS e installare il plug-in del dispositivo NVIDIA.

EKS Auto Mode

Crea il cluster EKS Auto Mode

```
eksctl create cluster \
  --name=$CLUSTER_NAME \
  --region=$AWS_REGION \
  --enable-auto-mode \
  --version=1.36 \
  --zones=$AZS
```

Il completamento di questo comando richiede alcuni minuti. Dopo il completamento, aggiorna `eksctl` automaticamente il file kubeconfig in modo che funzioni con il cluster appena eseguito il provisioning. Verifica che il cluster sia operativo:

```
kubectl get pods --all-namespaces
```

Output previsto:

```
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   metrics-server-55cf976ddd-cz2mw   1/1     Running   0          3m
kube-system   metrics-server-55cf976ddd-wrjvv   1/1     Running   0          3m
```

In EKS Auto Mode, VPC CNI, kube-proxy e CoreDNS vengono eseguiti come componenti gestiti e non vengono visualizzati come pod in. `kube-system`

Crea l'impostazione predefinita StorageClass

I carichi di lavoro della GPU richiedono spesso uno storage persistente, ad esempio per memorizzare nella cache i pesi dei modelli in modo che Kubernetes non li scarichi nuovamente a ogni riavvio del pod. EKS Auto Mode include una funzionalità di archiviazione a blocchi integrata, quindi non è necessario installare il driver Amazon EBS CSI. Creane uno `StorageClass` che faccia riferimento al provisioner Auto Mode:

```
cat << EOF | kubectl apply -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```
metadata:
  name: gp3
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
EOF
```

Questo comando crea un gp3 `StorageClass` e lo contrassegna come predefinito del cluster con l'`storageclass.kubernetes.io/is-default-class`annotazione. Tutti quelli `PersistentVolumeClaim` che omettono lo `storageClassName` usano. `StorageClass` L'`volumeBindingMode: WaitForFirstConsumer`impostazione ritarda la creazione del volume fino a quando Kubernetes pianifica un pod, quindi il volume EBS arriva nella stessa zona di disponibilità del pod.

Self-managed Karpenter

Autentica Helm all'ECR pubblico

`eksctl`estrae il grafico Karpenter Helm da Amazon Public ECR. Effettua l'autenticazione prima di creare il cluster per evitare un errore 403 nella fase di installazione di Helm:

```
aws ecr-public get-login-password --region us-east-1 \
  | helm registry login --username AWS --password-stdin public.ecr.aws
```

Public ECR è un servizio globale ospitato in. `us-east-1` Usalo `--region us-east-1` qui indipendentemente dalla regione in cui si trova il tuo cluster EKS.

Risultato previsto: `Login Succeeded`

Crea il cluster EKS con Karpenter

Memorizza la tua versione di Karpenter in una variabile di ambiente per un uso successivo. Per le versioni più recenti di Karpenter, consulta le versioni di Karpenter su. [https://github.com/aws/karpenter-provider-aws/releases](https://github.com/aws/karpenter-provider-aws/releases) GitHub

```
export KARPENTER_VERSION=1.14.0
```

Example Configurazione del cluster: YAML

```
cat << EOF > /tmp/cluster-karpenter.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: ${CLUSTER_NAME}
  region: ${AWS_REGION}
  version: "1.36"
  tags:
    karpenter.sh/discovery: ${CLUSTER_NAME}

availabilityZones: [$(echo $AZS | sed 's/,/, /g')]

autoModeConfig:
  enabled: false

iam:
  withOIDC: true

karpenter:
  version: "${KARPENTER_VERSION}"
  withSpotInterruptionQueue: true

managedNodeGroups:
  - name: system
    instanceTypes: ["m6i.2xlarge", "m5.2xlarge", "m6a.2xlarge", "m5a.2xlarge",
 "m7i.2xlarge"]
    desiredCapacity: 2
    minSize: 2
    maxSize: 3
    labels:
      node-role: system
    tags:
      karpenter.sh/discovery: ${CLUSTER_NAME}

addons:
  - name: eks-pod-identity-agent
  - name: eks-node-monitoring-agent
  - name: aws-ebs-csi-driver
EOF
```

Il gruppo di nodi `system` gestiti elenca diversi tipi di istanza della stessa dimensione (8 vCPU, 32 GiB) anziché un singolo tipo. Se un tipo è temporaneamente non disponibile in una zona di disponibilità, il gruppo Auto Scaling del gruppo di nodi ricorre a un altro tipo, il che evita `InsufficientInstanceCapacity` errori di avvio. Per un gruppo di nodi On-Demand gestito, tutti i tipi di istanza elencati devono avere la stessa vCPU e memoria in modo che la dimensione del nodo sia uniforme.

Crea il cluster con il file di configurazione:

```
eksctl create cluster -f /tmp/cluster-karpenter.yaml
```

Questo comando richiede circa 15 minuti. Crea un cluster EKS con un gruppo di nodi gestiti dedicato all'hosting di componenti aggiuntivi e il controller Karpenter. Karpenter è installato con la coda di interruzione Spot abilitata in modo da poter gestire i consigli relativi alle interruzioni e al ribilanciamento di Spot. L'`autoModeConfig.enabled: false`impostazione rende esplicito che questo cluster non utilizza la modalità automatica EKS, pertanto i componenti Karpenter installati in questo percorso sono responsabili della gestione dei nodi.

Il cluster ottiene anche l'EKS Pod Identity Agent, l'agente di monitoraggio dei nodi EKS e il driver Amazon EBS CSI installati come componenti aggiuntivi EKS. EKS Pod Identity viene utilizzato più avanti nella guida. L'agente di monitoraggio dei nodi EKS viene eseguito su ogni nodo e legge i log del kernel per impostare condizioni del nodo `AcceleratedHardwareReady``KernelReady`, ad esempio`NetworkingReady`, e, che Karpenter utilizza per decidere quando sostituire un nodo non integro.

Verifica che il cluster sia operativo:

```
kubectl get pods --all-namespaces
```

L'output previsto include Karpenter, CoreDNS, kube-proxy, aws-node (VPC CNI), EKS Pod Identity Agent e l'agente di monitoraggio dei nodi EKS.

```
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
karpenter     karpenter-567547464c-s6vkx          1/1     Running   0          3m40s
karpenter     karpenter-567547464c-x7gmw          1/1     Running   0          3m40s
kube-system   aws-node-b6gf2                      2/2     Running   0          12m
kube-system   aws-node-lcphh                      2/2     Running   0          12m
kube-system   coredns-7d4dcbf4fb-ccvrr            1/1     Running   0          16m
kube-system   coredns-7d4dcbf4fb-qbhk2            1/1     Running   0          16m
```

```
kube-system   eks-node-monitoring-agent-h79vm   1/1     Running   0          9m45s
kube-system   eks-node-monitoring-agent-tf4dw   1/1     Running   0          9m45s
kube-system   eks-pod-identity-agent-5jbtc      1/1     Running   0          12m
kube-system   eks-pod-identity-agent-rwcrc      1/1     Running   0          12m
kube-system   kube-proxy-p4bmq                  1/1     Running   0          12m
kube-system   kube-proxy-v5nwr                  1/1     Running   0          12m
kube-system   metrics-server-5b966ff79c-hr58p   1/1     Running   0          9m22s
kube-system   metrics-server-5b966ff79c-szs2d   1/1     Running   0          9m22s
```

Abilita le porte delle funzionalità di Karpenter

La modalità automatica EKS consente la riparazione automatica dei nodi e la capacità statica per impostazione predefinita. Su Karpenter autogestito, è necessario abilitare esplicitamente il `NodeRepair` feature gate e il `StaticCapacity` gate for static NodePools, che mantengono un numero fisso di nodi `spec.replicas` anziché ridimensionarsi in risposta ai pod in sospeso. Entrambi i gate sono Alpha e sono disattivati per impostazione predefinita in Karpenter.
Il comando seguente corregge la distribuzione di Karpenter per abilitare entrambi i gate. L'aggiornamento dell'ambiente di distribuzione attiva il rollout dei pod Karpenter:

```
kubectl set env deployment/karpenter -n karpenter \
  FEATURE_GATES=NodeRepair=true,StaticCapacity=true
```

Output previsto:

```
deployment.apps/karpenter env updated
```

Attendi il lancio dei pod Karpenter:

```
kubectl rollout status deployment/karpenter -n karpenter
```

Installa il plug-in del dispositivo NVIDIA

L'AMI EKS-optimized AL2023 non include il plug-in del dispositivo NVIDIA (a differenza dell'AMI Bottlerocket utilizzata da EKS Auto Mode). Installalo tramite Helm per rendere le risorse GPU utilizzabili con i Pods sul cluster.

```
helm repo add nvdp https://nvidia.github.io/k8s-device-plugin
helm repo update
```

```
cat << 'EOF' > /tmp/nvdp-values.yaml
```

```
mofedEnabled: false
nodeSelector:
  amiFamily: al2023
gfd:
  enabled: true
nfd:
  worker:
    tolerations:
      - operator: "Exists"
EOF
```

```
helm install nvidia-device-plugin nvdp/nvidia-device-plugin \
  --namespace kube-system \
  -f /tmp/nvdp-values.yaml
```

- `mofedEnabled: false`: disabilita il controllo di Mellanox OFED (), che non utilizza InfiniBand AWS
- `nodeSelector.amiFamily: al2023`: applica solo i DaemonSet nodi AL2023 (Bottlerocket ha già il plugin integrato)
- `gfd.enabled: true`: abilita le etichette GPU Feature Discovery (,, ecc.) `nvidia.com/gpu.product` `nvidia.com/gpu.memory`

Verifica che il plug-in del dispositivo NVIDIA sia installato. L'aspettativa è che non ci siano plug-in Pod per dispositivi finché non viene fornita una GPU NodePool con l'etichetta corrispondente.

```
kubectl get daemonset nvidia-device-plugin -n kube-system
```

Output previsto:

```
NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR      AGE
nvidia-device-plugin   0         0         0       0            0
 amiFamily=al2023   2m5s
```

Quindi, creane uno per uso generico. NodePool Il gruppo di nodi `system` gestiti esegue i componenti aggiuntivi e i controller. Aggiungi un Karpenter generico in modo che i carichi di lavoro non basati su GPU `NodePool` possano essere scalati su richiesta senza dover passare ai nodi. `system` La GPU viene creata più avanti nel passaggio 2`NodePool`. ???

### General-purpose EC2NodeClass e NodePool YAML

```
cat << EOF | kubectl apply -f -
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: general-purpose
spec:
  role: "eksctl-KarpenterNodeRole-${CLUSTER_NAME}"
  amiSelectorTerms:
    - alias: al2023@latest
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: ${CLUSTER_NAME}
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: ${CLUSTER_NAME}
  tags:
    karpenter.sh/discovery: ${CLUSTER_NAME}
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: general-purpose
spec:
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: general-purpose
      requirements:
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
        - key: karpenter.k8s.aws/instance-category
          operator: In
          values: ["c", "m", "r"]
        - key: karpenter.k8s.aws/instance-generation
          operator: Gt
          values: ["4"]
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
```

```
  limits:
    cpu: 1000
    memory: 1000Gi
EOF
```

Questo non NodePool ha contaminazioni, quindi qualsiasi Pod che non tollera la contaminazione della GPU può programmarlo.

Crea l'impostazione predefinita StorageClass

I carichi di lavoro della GPU richiedono spesso uno storage persistente, ad esempio per memorizzare nella cache i pesi dei modelli in modo che Kubernetes non li scarichi nuovamente a ogni riavvio del pod. Il driver Amazon EBS CSI è già installato dal `addons:` blocco nella configurazione del cluster all'inizio di questo passaggio, quindi devi solo creare: `StorageClass`

```
cat << EOF | kubectl apply -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gp3
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
EOF
```

Questo comando crea un cluster `gp3` `StorageClass` e lo contrassegna come predefinito del cluster con l'annotazione. `storageclass.kubernetes.io/is-default-class` Tutti quelli `PersistentVolumeClaim` che omettono lo `storageClassName` usano. `StorageClass` L'`volumeBindingMode: WaitForFirstConsumer`impostazione ritarda la creazione del volume fino a quando Kubernetes pianifica un pod, quindi il volume EBS arriva nella stessa zona di disponibilità del pod.

> **Warning**
>
> Sia per i percorsi EKS Auto Mode che per i percorsi Karpenter autogestiti, la riparazione automatica dei nodi si comporta allo stesso modo per i nodi forniti da. NodePools La

riparazione automatica dei nodi in EKS Auto Mode e Karpenter è un  potente metodo di interruzione che bypassa, annota e. PodDisruptionBudgets `karpenter.sh/do-not-disrupt` `terminationGracePeriod` La riparazione automatica dei nodi richiede 10 minuti prima di sostituire un nodo con la `AcceleratedHardwareReady` condizione impostata su e 30 minuti per le altre condizioni di `False` riparazione. https://docs.aws.amazon.com/eks/latest/userguide/node-repair.html

Configura il bilanciamento del carico

I passaggi successivi di questa guida espongono Grafana (e, nella procedura dettagliata di inferenza, un endpoint del modello) tramite un AWS Application Load Balancer (ALB). Configura ora i prerequisiti per il bilanciamento del carico in modo che tali sezioni possano crearne uno senza ulteriori configurazioni. `Ingress`

L'ALB viene inserito nelle sottoreti VPC in base ai tag delle risorse. AWS Le sottoreti pubbliche utilizzate per i load balancer connessi a Internet devono essere contrassegnate con = e le sottoreti private utilizzate per i load balancer interni devono essere contrassegnate con `kubernetes.io/role/elb` =1. `kubernetes.io/role/internal-elb` 1 Poiché hai creato questo cluster con, questi tag esistono già nelle sottoreti create. `eksctl` `eksctl` Se utilizzi il tuo VPC o le tue sottoreti importate, contrassegnale manualmente. Per informazioni dettagliate, vedi the section called "Assegnazione di tag alle sottoreti".

Crea il sistema di bilanciamento del carico IngressClass

Il resto della configurazione differisce in base al percorso. EKS Auto Mode include un controller ALB integrato, quindi puoi crearne uno `IngressClass` tu stesso. Self-managed Karpenter richiede l'installazione del AWS Load Balancer Controller con Helm, e il diagramma Helm lo crea per te. `IngressClass`

EKS Auto Mode

EKS Auto Mode include un controller ALB integrato, quindi non è necessario installare componenti aggiuntivi. `IngressClass`Create un nome `alb` che punti al controller integrato. Ciascuno `Ingress` creato successivamente fa riferimento a questa classe per nome e imposta il proprio schema e altre opzioni tramite annotazioni.

```
cat << EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
```

```
kind: IngressClass
metadata:
  name: alb
spec:
  controller: eks.amazonaws.com/alb
EOF
```

Self-managed Karpenter

Il AWS Load Balancer Controller non è un EKS-managed componente aggiuntivo, quindi non può entrare nel `addons:` blocco di configurazione del `eksctl` cluster come fa il driver EBS CSI. Installalo con Helm.

Scarica la policy IAM

Il controller necessita delle autorizzazioni IAM per creare e gestire gli ALB. Scarica il JSON della policy IAM ufficiale:

```
curl -o iam_policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/install/iam_policy.json
```

Crea la policy IAM:

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document file://iam_policy.json
```

Crea l'associazione Pod Identity

Il cluster installa già l'`eks-pod-identity-agent`add-on nel passaggio 1, quindi EKS Pod Identity è disponibile. Crea un'associazione Pod Identity che associ la policy IAM all'account di servizio del controller:

```
eksctl create podidentityassociation \
  --cluster ${CLUSTER_NAME} \
  --region ${AWS_REGION} \
  --namespace kube-system \
  --service-account-name aws-load-balancer-controller \
  --permission-policy-arns arn:aws:iam::$(aws sts get-caller-identity --query
 Account --output text):policy/AWSLoadBalancerControllerIAMPolicy
```

Installa con Helm

Aggiungi il repository Helm di EKS charts:

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update
```

Cerca l'ID VPC del tuo cluster. I valori Helm ne hanno bisogno affinché il controller possa scoprire le sottoreti:

```
VPC_ID=$(aws eks describe-cluster \
  --name ${CLUSTER_NAME} \
  --region ${AWS_REGION} \
  --query 'cluster.resourcesVpcConfig.vpcId' \
  --output text)
```

Installare il controller. La tolleranza `nodeSelector` e la tolleranza collocano il controller nel gruppo di nodi `system` gestiti accanto all'infrastruttura critica del cluster, anziché competere con le GPU o i nodi generici in termini di capacità: Karpenter-launched

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  --namespace kube-system \
  --version 3.4.2 \
  --set clusterName=${CLUSTER_NAME} \
  --set region=${AWS_REGION} \
  --set vpcId=${VPC_ID} \
  --set serviceAccount.create=true \
  --set serviceAccount.name=aws-load-balancer-controller \
  --set "nodeSelector.node-role=system" \
  --set "tolerations[0].key=CriticalAddonsOnly" \
  --set "tolerations[0].operator=Exists"
```

Verifica che il controller sia in funzione. Dovrebbe riportare due repliche disponibili:

```
kubectl get deployment aws-load-balancer-controller -n kube-system
```

Verifica il IngressClass

Il diagramma Helm crea un `IngressClass` nome `alb` come parte dell'installazione del controller, che punta al controller autogestito. `ingress.k8s.aws/alb` Non è necessario crearlo manualmente. Conferma che esiste:

```
kubectl get ingressclass alb
```

Output previsto:

```
NAME   CONTROLLER            PARAMETERS   AGE
alb    ingress.k8s.aws/alb   <none>       22s
```

Entrambi i percorsi ora hanno un `albIngressClass`. Le sezioni Grafana e Inference fanno riferimento `ingressClassName: alb` e impostano lo schema ALB, la lista consentita Source-IP e altri comportamenti tramite preannotazioni. `Ingress`

## Fase 2: Creare una GPU dinamica NodePool

Definisci una NodePool che esegua il provisioning dinamico delle istanze G-family GPU con una generazione superiore a 4 e inferiore a 7, utilizzando la capacità Spot con On-Demand come riserva. I percorsi EKS Auto Mode e Karpenter utilizzano entrambi la stessa NodePool API con l'unica differenza che fa riferimento. NodeClass In EKS Auto Mode, il bundle seleziona `default` NodeClass già l'AMI corretta e configura il pull parallelo SOCI, quindi NodePool è l'unico oggetto che si crea. In Karpenter autogestito, è inoltre necessaria una personalizzazione `EC2NodeClass` che aggiunga l'AMI e ottimizzi il SOCI.

> **Important**
>
> Il tipo di istanza G7 EC2 richiede il driver NVIDIA versione 595 o successiva. Le AMI EKS-optimized accelerate attualmente includono il driver NVIDIA versione 580, che non supporta le istanze G7. NodePool In questo passaggio vincola la generazione dell'istanza a meno di 7 in modo che Karpenter non selezioni un'istanza G7. Per utilizzare le istanze G7 con Amazon EKS, devi creare un'AMI personalizzata con il driver NVIDIA versione 595. Per ulteriori informazioni, consulta the section called “Usa le AMI GPU EKS Linux”.

EKS Auto Mode

Nella modalità EKS Auto, il pacchetto seleziona `default` NodeClass automaticamente l'AMI Bottlerocket per le istanze GPU, che include i driver NVIDIA preinstallati, il plug-in del dispositivo NVIDIA e il pull parallelo SOCI. È NodePool sufficiente applicare `default` NodeClass un codice che faccia riferimento a:

```
cat << 'EOF' | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-inf
spec:
  template:
    metadata:
      labels:
        guide: ai-eks-docs
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      requirements:
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["spot", "on-demand"]
        - key: eks.amazonaws.com/instance-category
          operator: In
          values: ["g"]
        - key: eks.amazonaws.com/instance-generation
          operator: Gt
          values: ["4"]
        - key: eks.amazonaws.com/instance-generation
          operator: Lt
          values: ["7"]
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
  limits:
    cpu: 1000
    memory: 5000Gi
EOF
```

Ciò NodePool fornisce istanze G-family GPU con una generazione superiore a 4 e inferiore a 7 (G5, G6e, ecc.). EKS Auto Mode sceglie un tipo di istanza all'interno di questi vincoli.

Self-managed Karpenter

Self-managed Karpenter non include un valore predefinito. NodeClass Per prima cosa si crea un modulo `EC2NodeClass` che inserisce l'alias EKS-optimized NVIDIA AL2023 AMI, abilita SOCI tramite il `FastImagePull` feature gate e si configura `instanceStorePolicy: RAID0` per spostare la cache delle immagini contenuta su NVMe locali. Quindi si crea quello NodePool che fa riferimento ad esso.

Crea il EC2NodeClass

Example

```
cat << EOF | kubectl apply -f -
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-inf
  labels:
    guide: ai-eks-docs
spec:
  role: "eksctl-KarpenterNodeRole-${CLUSTER_NAME}"
  amiSelectorTerms:
    - alias: al2023@latest
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: ${CLUSTER_NAME}
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: ${CLUSTER_NAME}
  tags:
    karpenter.sh/discovery: ${CLUSTER_NAME}
  instanceStorePolicy: RAID0
  userData: |
    apiVersion: node.eks.aws/v1alpha1
    kind: NodeConfig
    spec:
      featureGates:
        FastImagePull: true
EOF
```

`instanceStorePolicy: RAID0`assembla i dischi NVMe locali in un array. RAID-0 L'alias `al2023@latest` AMI si risolve nell'AMI AL2023. EKS-optimized Quando Karpenter avvia un tipo

di istanza GPU, seleziona automaticamente la variante accelerata AL2023_x86_64_NVIDIA, che include il driver NVIDIA preinstallato.

Il feature gate abilita la modalità pull parallela SOCI snapshotter, che scarica e decomprime i livelli di immagine contemporaneamente. `FastImagePull` Ciò corrisponde al comportamento della modalità automatica EKS sulle famiglie di istanze G, P e Trn.

Per ulteriori opzioni di ottimizzazione SOCI (download simultanei per immagine, dimensione dei blocchi, ecc.), consultate il progetto SOCI di Karpenter. https://github.com/aws-samples/karpenter-blueprints/tree/main/blueprints/soci-snapshotter

EC2NodeClassConvalida:

```
kubectl get ec2nodeclass gpu-inf
```

Risultato previsto:`READY True`. Se`False`, esegui `kubectl describe ec2nodeclass gpu-inf` e controlla le condizioni per i tag di sottorete o gruppo di sicurezza mancanti.

Crea la GPU NodePool

Example

```
cat << EOF | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-inf
spec:
  template:
    metadata:
      labels:
        guide: ai-eks-docs
        amiFamily: al2023
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: gpu-inf
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      requirements:
        - key: karpenter.sh/capacity-type
```

```
          operator: In
          values: ["spot", "on-demand"]
        - key: karpenter.k8s.aws/instance-category
          operator: In
          values: ["g"]
        - key: karpenter.k8s.aws/instance-generation
          operator: Gt
          values: ["4"]
        - key: karpenter.k8s.aws/instance-generation
          operator: Lt
          values: ["7"]
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
  limits:
    cpu: 1000
    memory: 5000Gi
EOF
```

L'`amiFamily: al2023`etichetta sul modello di nodo è quella DaemonSet utilizzata dal plug-in del dispositivo NVIDIA per selezionare questi nodi. Karpenter sceglie un tipo di istanza all'interno di questi vincoli. La `nvidia.com/gpu:NoSchedule` contaminazione garantisce che solo i GPU-eligible Pod siano pianificati su questi nodi.

Convalida che è stato creato NodePool :

```
kubectl get nodepool gpu-inf
```

Output previsto:

```
NAME      NODECLASS   NODES   READY   AGE
gpu-inf   default     0       True    8s
```

Nel percorso Karpenter autogestito, la colonna NODECLASS mostra invece di. `gpu-inf` `default`

## Fase 3: Esegui il test con un Pod di esempio

Verifica la NodePool configurazione della GPU con un `nvidia-smi` Pod.

```
cat << EOF | kubectl apply -f -
apiVersion: v1
```

```
kind: Pod
metadata:
  name: nvidia-smi
  labels:
    guide: ai-eks-docs
spec:
  tolerations:
    - key: "nvidia.com/gpu"
      operator: "Exists"
      effect: "NoSchedule"
  containers:
    - name: nvidia-smi
      image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
      command: ["nvidia-smi"]
      resources:
        limits:
          nvidia.com/gpu: 1
  restartPolicy: OnFailure
EOF
```

Verifica che il Pod sia pianificato e completato correttamente.

```
kubectl get pods
```

Output previsto:

```
NAME         READY   STATUS      RESTARTS   AGE
nvidia-smi   0/1     Completed   0          67s
```

Lo STATO: Completato indica che il comando nvidia-smi è stato eseguito ed è terminato. Controlla i log del Pod per vedere la GPU rilevata dal nodo.

```
kubectl logs nvidia-smi
```

Output previsto:

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 580.159.03             Driver Version: 580.159.03     CUDA Version: 13.0
   |
+-----------------------------------------+------------------------
+----------------------+
```

```
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
 ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute
 M. |
|                                         |                        |               MIG
 M. |
|=========================================+========================
+======================|
|   0  NVIDIA L4                      On  |   00000000:35:00.0 Off |
 0 |
| N/A   42C    P8             16W /   72W |       0MiB /  23034MiB |      0%
 Default |
|                                         |                        |
 N/A |
+-----------------------------------------+------------------------
+----------------------+
```

L'output mostra il modello della GPU, la versione del driver, la versione CUDA e la memoria disponibile. In questo esempio, Karpenter ha effettuato il provisioning di un'istanza G6 con una GPU NVIDIA L4 con 24 GB di memoria. 42°C è la temperatura attuale della GPU e P8 è uno stato di inattività a basso consumo, previsto perché nessun carico di lavoro utilizza la GPU. I valori da 16 W/72 W mostrano l'assorbimento di corrente rispetto alla capacità massima e 0 MiB/23034 MiB mostra la memoria GPU attualmente utilizzata rispetto al totale disponibile. Poiché il Pod ha appena eseguito nvidia-smi ed è uscito, la GPU non utilizza alcun carico di lavoro, quindi la memoria è a 0 e l'alimentazione è inattiva. La versione del driver della GPU NVIDIA (580.159.03) proviene dall'AMI Bottlerocket, mentre la versione CUDA (13.0) proviene dall'immagine del contenitore. Il modello e la memoria della GPU varieranno a seconda del tipo di istanza selezionato da Karpenter. Le istanze G5 hanno GPU NVIDIA A10G (24 GB), le istanze G6 hanno GPU NVIDIA L4 (24 GB) e le istanze G6e hanno GPU NVIDIA L40S (48 GB).

Per capire come Karpenter e lo scheduler Kubernetes si sono coordinati per fornire un nodo e posizionare il Pod, controlla gli eventi del ciclo di vita del Pod:

```
kubectl describe po nvidia-smi
```

Output previsto:

```
Events:
  Type     Reason                  Age   From                    Message
  ----     ------                  ----  ----                    -------
```

```
  Warning  FailedScheduling        60s   default-scheduler      0/2 nodes are
 available: 2 node(s) had untolerated taint(s). no new claims to deallocate,
 preemption: 0/2 nodes are available: 2 Preemption is not helpful for scheduling.
  Normal   Nominated               59s   eks-auto-mode/compute  Pod should schedule on:
 nodeclaim/gpu-inf-vxcnj
  Normal   Scheduled               24s   default-scheduler      Successfully assigned
 default/nvidia-smi to i-0fb17a09bc4203164
  Warning  FailedCreatePodSandBox  21s   kubelet                Failed to create
 pod sandbox: rpc error: code = Unknown desc = failed to setup network for sandbox
 "7f85e25b220c8fb245187758dbbbc8efb3d40f3e49e13054404880daf4c3b2f0": plugin type="aws-
cni" name="aws-cni" failed (add): add cmd: failed to setup network policy
  Normal   Pulling                  7s   kubelet                spec.containers{nvidia-
smi}: Pulling image "public.ecr.aws/amazonlinux/amazonlinux:2023-minimal"
  Normal   Pulled                   5s   kubelet                spec.containers{nvidia-
smi}: Successfully pulled image "public.ecr.aws/amazonlinux/amazonlinux:2023-minimal"
 in 1.237s (1.237s including waiting). Image size: 37442701 bytes.
  Normal   Created                  5s   kubelet                spec.containers{nvidia-
smi}: Container created
  Normal   Started                  5s   kubelet                spec.containers{nvidia-
smi}: Container started
```

Questi eventi mostrano la sequenza di pianificazione del Pod: il Pod inizialmente non riesce a pianificare perché non esistono nodi GPU (FailedScheduling), Karpenter ne nomina un nuovo NodeClaim (Nominato), lo scheduler assegna il Pod una volta che il nodo è pronto (Scheduled), quindi l'immagine del contenitore viene estratta e avviata. EKS Auto Mode viene fornito con il pull parallelo SOCI (Seekable OCI) installato e configurato immediatamente sulle istanze G, P e Trn. Nota: grazie al pull parallelo SOCI, l'immagine del contenitore è stata estratta dall'ECR in meno di 2 secondi (1,237 secondi).

A NodeClaim è una richiesta creata da Karpenter per eseguire il provisioning di un nodo specifico. Mostra il tipo di istanza, il tipo di capacità, l'AZ e se il nodo è pronto.

```
kubectl get nodeclaims
```

NodeClaim Risultato previsto:

```
NAME            TYPE         CAPACITY   ZONE         NODE                  READY   AGE
gpu-inf-vxcnj   g6.4xlarge   spot       us-east-2c   i-0fb17a09bc4203164   True    51s
```

Il tipo di istanza e l'AZ possono variare. Qualsiasi G-family istanza con una generazione superiore a 4 e inferiore a 7 è idonea.

L'`FailedCreatePodSandBox`avviso in arrivo `kubectl describe pod nvidia-smi` è temporaneo e previsto. Il VPC CNI si inizializza in modo asincrono dopo l'unione del nodo e il kubelet riprova automaticamente. Se il Pod rimane acceso, controlla gli eventi del nodo con. `ContainerCreating` `kubectl describe node <node-name>`

> **Tip**
>
> Se non viene visualizzato alcun nodo, verifica la presenza di errori di capacità insufficiente:
>
> ```
> kubectl get events | grep InsufficientCapacityError
> ```
>
> Karpenter memorizza nella cache le offerte non disponibili per 3 minuti. L'ampliamento dei tipi di istanze e degli AZ consentiti NodePool aumenta le possibilità di atterraggio.

> **Note**
>
> Le istanze Spot lanciate da Karpenter non verranno visualizzate nella console Spot Requests di EC2. Karpenter utilizza l'API EC2 con. `CreateFleet` `type: instant` Le istanze vengono visualizzate nella console EC2 Instances con un ciclo di vita. `spot`

## Fase 4: Aggiungere capacità riservata a (opzionale) NodePool

Per utilizzare innanzitutto la capacità riservata con Spot/On-Demand riserva, crea un On-Demand Capacity Reservation (ODCR) e collegalo al tuo NodeClass, quindi aggiorna la dinamica NodePool del passaggio 2 per consentire `reserved` anche la capacità. La chiamata all'API di prenotazione è la stessa per entrambi i percorsi; l' NodeClass allegato è diverso perché EKS Auto Mode e Karpenter autogestito ne utilizzano tipi diversi. NodeClass

> **Warning**
>
> Il comando seguente comporta un addebito per il tipo di istanza riservata fino a quando non viene annullato con. `aws ec2 cancel-capacity-reservation --capacity-reservation-id <id>`

Crea la prenotazione della capacità:

```
CR_AZ="us-east-2a"
INSTANCE_TYPE="g6e.4xlarge"

aws ec2 create-capacity-reservation \
  --instance-type $INSTANCE_TYPE \
  --instance-platform Linux/UNIX \
  --availability-zone "$CR_AZ" \
  --instance-count 1 \
  --instance-match-criteria open \
  --end-date-type unlimited
```

Se ricevi un `InsufficientInstanceCapacity` errore, passa `CR_AZ` a un altro AZ e riprova.

Cerca l'ID di prenotazione della capacità e memorizzalo in una variabile di shell per i seguenti passaggi:

```
CAPACITY_RESERVATION_ID=$(aws ec2 describe-capacity-reservations \
  --filters "Name=state,Values=active" "Name=instance-type,Values=${INSTANCE_TYPE}" \
  --query 'CapacityReservations[0].CapacityReservationId' \
  --output text \
  --region ${AWS_REGION})
echo "Capacity reservation ID: ${CAPACITY_RESERVATION_ID}"
```

Quindi applica le NodePool modifiche NodeClass e al tuo percorso:

EKS Auto Mode

In EKS Auto Mode, il pacchetto `default` NodeClass è di sola lettura, quindi crea una personalizzazione NodeClass che faccia riferimento alla prenotazione, quindi aggiorna il NodePool punto in alto NodeClass e aggiungi `reserved` capacità all'elenco. `capacity-type`

Example YAML personalizzato NodeClass

```
NODE_ROLE=$(kubectl get nodeclass default -o jsonpath='{.spec.role}')

cat << EOF | kubectl apply -f -
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu-inf
  labels:
```

```
    guide: ai-eks-docs
spec:
  role: "$NODE_ROLE"
  subnetSelectorTerms:
    - tags:
        alpha.eksctl.io/cluster-name: "$CLUSTER_NAME"
        kubernetes.io/role/internal-elb: "1"
  securityGroupSelectorTerms:
    - tags:
        aws:eks:cluster-name: "$CLUSTER_NAME"
  capacityReservationSelectorTerms:
    - id: "$CAPACITY_RESERVATION_ID"
EOF
```

Il `kubernetes.io/role/internal-elb: "1"` tag garantisce l'avvio dei nodi solo in sottoreti private.

Aggiorna il NodePool file per utilizzare ODCR-backed NodeClass e includere `reserved` come tipo di capacità:

Example NodePool YAML aggiornato

```
cat << EOF | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-inf
spec:
  template:
    metadata:
      labels:
        guide: ai-eks-docs
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: gpu-inf
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      requirements:
        - key: karpenter.sh/capacity-type
          operator: In
```

```
          values: ["spot", "on-demand", "reserved"]
        - key: eks.amazonaws.com/instance-category
          operator: In
          values: ["g"]
        - key: eks.amazonaws.com/instance-generation
          operator: Gt
          values: ["4"]
        - key: eks.amazonaws.com/instance-generation
          operator: Lt
          values: ["7"]
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
  limits:
    cpu: 1000
    memory: 5000Gi
EOF
```

Self-managed Karpenter

Per Karpenter autogestito, riapplica quello che `EC2NodeClass` hai creato nel passaggio 2 con l'aggiunta. `capacityReservationSelectorTerms` Il nome e la forma del campo corrispondono alla modalità automatica EKS `NodeClass` mostrata nell'altra scheda.

Example EC2NodeClass YAML aggiornato

```
cat << EOF | kubectl apply -f -
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-inf
  labels:
    guide: ai-eks-docs
spec:
  role: "eksctl-KarpenterNodeRole-${CLUSTER_NAME}"
  amiSelectorTerms:
    - alias: al2023@latest
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: ${CLUSTER_NAME}
  securityGroupSelectorTerms:
    - tags:
```

```
        karpenter.sh/discovery: ${CLUSTER_NAME}
  tags:
    karpenter.sh/discovery: ${CLUSTER_NAME}
  instanceStorePolicy: RAID0
  capacityReservationSelectorTerms:
    - id: "$CAPACITY_RESERVATION_ID"
  userData: |
    apiVersion: node.eks.aws/v1alpha1
    kind: NodeConfig
    spec:
      featureGates:
        FastImagePull: true
EOF
```

L'unica modifica rispetto al passaggio 2 è il nuovo `capacityReservationSelectorTerms` campo. Tutti gli altri campi rimangono invariati.

Aggiorna il NodePool file per includere `reserved` come tipo di capacità:

Example NodePool YAML aggiornato

```
cat << EOF | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-inf
spec:
  template:
    metadata:
      labels:
        guide: ai-eks-docs
        amiFamily: al2023
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: gpu-inf
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      requirements:
        - key: karpenter.sh/capacity-type
          operator: In
```

```
          values: ["spot", "on-demand", "reserved"]
        - key: karpenter.k8s.aws/instance-category
          operator: In
          values: ["g"]
        - key: karpenter.k8s.aws/instance-generation
          operator: Gt
          values: ["4"]
        - key: karpenter.k8s.aws/instance-generation
          operator: Lt
          values: ["7"]
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
  limits:
    cpu: 1000
    memory: 5000Gi
EOF
```

Karpenter considera l'opzione più conveniente `reserved` e la lancia per prima. Una volta che la prenotazione è completa, torna a Spot o. On-Demand

Verifica la priorità riservata e il fallback di Spot

Dopo aver applicato le modifiche, verifica che Karpenter dia la priorità alla capacità riservata e ritorni a Spot o. On-Demand Implementa una distribuzione con 2 repliche che richiede 1 GPU per pod. L'ODCR è per 1 istanza, quindi il primo Pod attiva Karpenter per avviare un nodo riservato. Il secondo Pod non può entrare nel nodo riservato e fa sì che Karpenter avvii un altro nodo da Spot o Capacity. On-Demand

```
cat << 'EOF' | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-overflow-test
  labels:
    guide: ai-eks-docs
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gpu-overflow-test
  template:
```

```
    metadata:
      labels:
        app: gpu-overflow-test
        guide: ai-eks-docs
    spec:
      tolerations:
        - key: nvidia.com/gpu
          operator: Exists
          effect: NoSchedule
      containers:
        - name: nvidia-smi
          image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
          command: ["sh", "-c", "nvidia-smi && sleep infinity"]
          resources:
            limits:
              nvidia.com/gpu: 1
EOF
```

A differenza del Pod di `nvidia-smi` test di Step 3, che è stato eseguito e chiuso, questo Deployment mantiene i Pod in funzione (`sleep infinity`) in modo che trattengano la GPU e non rilasciino il nodo.

Verifica i Pod pianificati su diversi nodi:

```
kubectl get pods -l app=gpu-overflow-test -o wide
```

Output previsto:

```
NAME                                 READY   STATUS    RESTARTS   AGE     IP
     NODE                   NOMINATED NODE   READINESS GATES
gpu-overflow-test-59b97944fb-lq56c   1/1     Running   0          2m42s
 192.168.186.240   i-057692590480155da   <none>           <none>
gpu-overflow-test-59b97944fb-z4zcx   1/1     Running   0          2m42s
 192.168.130.64    i-0521ecd1849fa0578   <none>           <none>
```

I due pod sono in esecuzione, ciascuno su un nodo diverso.

Seleziona NodeClaims per vedere i tipi di capacità:

```
kubectl get nodeclaims
```

Output previsto:

```
NAME            TYPE          CAPACITY    ZONE         NODE                  READY
 AGE
gpu-inf-shg5w   g6e.xlarge    reserved    us-east-2a   i-0ea91fdeef65b8cb6   True
 2m2s
gpu-inf-ssnqf   g6e.2xlarge   spot        us-east-2b   i-00ccf7ce65cf3f6ca   True
 112s
```

Il nodo riservato viene avviato per primo, seguito da uno Spot o da un On-Demand nodo una volta completata la prenotazione.

Pulisci la distribuzione di prova:

```
kubectl delete deployment gpu-overflow-test
```

## Monitoraggio

Installa uno stack di monitoraggio che raccolga le metriche di cluster, nodi e GPU in Amazon Managed Service for Prometheus (AMP) e visualizzale con Grafana. Il grafico Helm kube-prometheus-stack utilizza Prometheus per lo scraping e la scrittura remota delle metriche in AMP, oltre a un Grafana autogestito per le dashboard. NVIDIA DCGM Exporter aggiunge metriche (utilizzo, memoria, temperatura, potenza, NVLink, attività tensoriale). GPU-specific

Prometheus, Grafana e l'operatore atterrano su nodi non GPU per impostazione predefinita perché i nodi GPU sono portatori della contaminazione. `nvidia.com/gpu:NoSchedule` Node-exporter e il DCGM Exporter funzionano entrambi su nodi GPU, in modo da poter analizzare le metriche relative a host e GPU a livello di parco macchine.

Se hai aperto un nuovo terminale, imposta il nome e la regione del cluster:

```
export CLUSTER_NAME=ai-eks-docs
export AWS_REGION=us-east-2
```

Crea lo spazio di lavoro AMP

Crea uno spazio di lavoro AMP per archiviare le metriche:

```
aws amp create-workspace \
```

```
  --alias "amp-ws-${CLUSTER_NAME}" \
  --region ${AWS_REGION}
```

Ottieni l'ID dello spazio di lavoro:

```
AMP_WORKSPACE_ID=$(aws amp list-workspaces \
  --alias "amp-ws-${CLUSTER_NAME}" \
  --query 'workspaces[0].workspaceId' \
  --output text \
  --region ${AWS_REGION})

echo "AMP Workspace ID: ${AMP_WORKSPACE_ID}"
```

Ottieni l'endpoint di scrittura remota:

```
AMP_ENDPOINT=$(aws amp describe-workspace \
  --workspace-id ${AMP_WORKSPACE_ID} \
  --query 'workspace.prometheusEndpoint' \
  --output text \
  --region ${AWS_REGION})

echo "AMP Endpoint: ${AMP_ENDPOINT}"
```

Crea policy IAM e associazioni EKS Pod Identity

Crea una policy IAM che consenta a Prometheus di scrivere metriche in remoto e a Grafana di interrogarle:

```
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)

AMP_POLICY_ARN=$(aws iam create-policy \
  --policy-name "${CLUSTER_NAME}-amp-grafana-policy" \
  --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Sid\":
 \"AllowAMPReadWrite\", \"Effect\": \"Allow\", \"Action\": [\"aps:ListWorkspaces
\", \"aps:DescribeWorkspace\", \"aps:GetMetricMetadata\", \"aps:GetSeries
\", \"aps:QueryMetrics\", \"aps:RemoteWrite\", \"aps:GetLabels\"],
 \"Resource\": \"arn:aws:aps:${AWS_REGION}:${ACCOUNT_ID}:workspace/*\"},
 {\"Sid\": \"AllowCloudWatchMetrics\", \"Effect\": \"Allow\", \"Action
\": [\"cloudwatch:DescribeAlarmsForMetric\", \"cloudwatch:ListMetrics\",
 \"cloudwatch:GetMetricData\", \"cloudwatch:GetMetricStatistics\"], \"Resource\": \"*
\"}]}" \
  --query 'Policy.Arn' \
```

```
  --output text)

echo "AMP Policy ARN: ${AMP_POLICY_ARN}"
```

Crea il namespace di monitoraggio e gli account di servizio per Prometheus e Grafana:

```
kubectl create namespace monitoring
kubectl create serviceaccount amp-iamproxy-ingest-service-account -n monitoring
kubectl create serviceaccount grafana-sa -n monitoring
```

Crea EKS Pod Identity Associations per collegare gli account di servizio alla policy IAM:

```
eksctl create podidentityassociation \
  --cluster ${CLUSTER_NAME} \
  --namespace monitoring \
  --service-account-name amp-iamproxy-ingest-service-account \
  --role-name "${CLUSTER_NAME}-amp-ingest-role" \
  --permission-policy-arns ${AMP_POLICY_ARN} \
  --region ${AWS_REGION}

eksctl create podidentityassociation \
  --cluster ${CLUSTER_NAME} \
  --namespace monitoring \
  --service-account-name grafana-sa \
  --role-name "${CLUSTER_NAME}-grafana-role" \
  --permission-policy-arns ${AMP_POLICY_ARN} \
  --region ${AWS_REGION}
```

Verifica che entrambe le associazioni EKS Pod Identity siano state create:

```
eksctl get podidentityassociation --cluster ${CLUSTER_NAME} --region ${AWS_REGION}
```

L'output previsto dovrebbe includere entrambi `amp-iamproxy-ingest-service-account` e `grafana-sa` nel `monitoring` namespace.

Installa kube-prometheus-stack

Aggiungi il repository Helm:

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
helm repo update
```

Questo file di valori omette un NodeSelector per Prometheus, Grafana e l'operatore: la contaminazione dei nodi GPU li tiene lontani dai nodi della GPU, quindi per impostazione predefinita `nvidia.com/gpu:NoSchedule` finiscono nel sistema o nel pool generico. Node-exporter utilizza una tolleranza jolly, quindi viene eseguito su tutti i nodi, inclusi i nodi GPU, per raccogliere le metriche a livello di flotta.

Il file dei valori configura anche un Grafana in `Ingress` modo che il grafico fornisca un ALB rivolto a Internet per l'accesso al browser, utilizzando quello creato in Set up load balancing. `alb` `IngressClass` ??? L'`alb.ingress.kubernetes.io/inbound-cidrs`annotazione limita il load balancer al tuo indirizzo IP, quindi impostalo prima dell'installazione.

> ⚠ Grafana è accessibile pubblicamente tramite HTTP con credenziali predefinite
>
> Grafana include le credenziali di amministratore predefinite e fornisce l'accesso amministrativo completo ai dati di monitoraggio. Un sistema di bilanciamento del carico con connessione a Internet colloca questa interfaccia sulla rete Internet pubblica tramite HTTP in testo semplice. Gli scanner automatici scoprono i sistemi di bilanciamento del carico pubblici in pochi minuti. È necessario limitare l'accesso con l'`alb.ingress.kubernetes.io/inbound-cidrs`annotazione e considerare l'elenco degli indirizzi IP di origine come una protezione minima, non completa. Cambia anche la password amministratore predefinita di Grafana. Per una postura più sicura, utilizza uno schema interno (raggiungibile solo dall'interno del tuo VPC o da una VPN connessa) impostando lo schema su e aggiungi un certificato `internal` TLS.

Trova il tuo indirizzo IP pubblico e memorizzalo come CIDR. `/32` Il file dei valori legge questo dalla variabile di `MY_CIDR` ambiente:

```
export MY_CIDR="$(curl -s https://checkip.amazonaws.com)/32"
echo $MY_CIDR
```

Il risultato è simile`203.0.113.4/32`. Se la tua rete assegna gli indirizzi in modo dinamico, il tuo indirizzo IP potrebbe cambiare nel tempo, quindi valuta la possibilità di impostare un intervallo più ampio come. `203.0.113.0/24`

Crea il file dei valori:

Example file di valori kube-prometheus-stack

```
cat << EOF > /tmp/kube-prometheus-values.yaml
alertmanager:
  enabled: false

prometheus-adapter:
  enabled: false

prometheus:
  serviceAccount:
    create: false
    name: amp-iamproxy-ingest-service-account
  prometheusSpec:
    serviceAccountName: amp-iamproxy-ingest-service-account
    enableRemoteWriteReceiver: true
    retention: 2h
    scrapeInterval: 30s
    evaluationInterval: 30s
    podMonitorSelectorNilUsesHelmValues: false
    serviceMonitorSelectorNilUsesHelmValues: false
    resources:
      requests:
        cpu: 500m
        memory: 1Gi
      limits:
        memory: 8Gi
    remoteWrite:
      - url: "${AMP_ENDPOINT}api/v1/remote_write"
        sigv4:
          region: "${AWS_REGION}"
        queueConfig:
          maxSamplesPerSend: 1000
          maxShards: 200
          capacity: 2500

prometheusOperator:
  resources:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      memory: 256Mi
```

```
kube-state-metrics:
  resources:
    requests:
      cpu: 50m
      memory: 128Mi
    limits:
      memory: 512Mi

grafana:
  enabled: true
  ingress:
    enabled: true
    ingressClassName: alb
    labels:
      guide: ai-eks-docs
    annotations:
      alb.ingress.kubernetes.io/scheme: internet-facing
      alb.ingress.kubernetes.io/target-type: ip
      alb.ingress.kubernetes.io/inbound-cidrs: ${MY_CIDR}
      alb.ingress.kubernetes.io/load-balancer-name: grafana-ai-eks-docs
      alb.ingress.kubernetes.io/healthcheck-path: /api/health
  serviceAccount:
    create: false
    name: grafana-sa
  resources:
    requests:
      cpu: 100m
      memory: 256Mi
    limits:
      memory: 1Gi
  grafana.ini:
    auth.sigv4:
      enabled: true
  sidecar:
    datasources:
      defaultDatasourceEnabled: false
  plugins:
    - grafana-amazonprometheus-datasource
  additionalDataSources:
    - name: Amazon-Managed-Prometheus
      type: grafana-amazonprometheus-datasource
      access: proxy
      url: "${AMP_ENDPOINT}"
      isDefault: true
```

```
      jsonData:
        sigV4Auth: true
        defaultRegion: "${AWS_REGION}"
        sigV4Region: "${AWS_REGION}"
      editable: true
  dashboardProviders:
    dashboardproviders.yaml:
      apiVersion: 1
      providers:
        - name: default
          orgId: 1
          folder: 'GPU Monitoring'
          type: file
          disableDeletion: false
          editable: true
          options:
            path: /var/lib/grafana/dashboards/default
  dashboards:
    default:
      nvidia-dcgm:
        gnetId: 25261
        revision: 1
        datasource:
          - name: DS_PROMETHEUS
            value: Amazon-Managed-Prometheus
      vllm:
        gnetId: 25263
        revision: 1
        datasource:
          - name: DS_PROMETHEUS
            value: Amazon-Managed-Prometheus
      vllm-load-analysis:
        gnetId: 25494
        revision: 1
        datasource:
          - name: DS_PROMETHEUS
            value: Amazon-Managed-Prometheus

prometheus-node-exporter:
  resources:
    requests:
      cpu: 50m
      memory: 64Mi
    limits:
```

```
      memory: 128Mi
  tolerations:
    - operator: Exists
EOF
```

Convalida che le variabili siano state compilate correttamente:

```
grep -E "url:|region:" /tmp/kube-prometheus-values.yaml
```

Dovresti vedere l'URL completo dell'endpoint AMP (che inizia con`https://aps-workspaces…` ) e la tua regione. Se una delle due è vuota, riesporta le variabili e ricrea il file.

Installa il grafico:

```
helm install kube-prometheus-stack prometheus-community/kube-prometheus-stack \
  --namespace monitoring \
  -f /tmp/kube-prometheus-values.yaml
```

Verifica che i pod siano in funzione:

```
kubectl get pods -n monitoring
```

Output previsto:

```
NAME                                                     READY   STATUS    RESTARTS
 AGE
kube-prometheus-stack-grafana-7c58f54f77-rftrj           3/3     Running   0
 4m
kube-prometheus-stack-kube-state-metrics-d68dcbc84-5smxq 1/1     Running   0
 4m
kube-prometheus-stack-operator-5895df479f-ttm47          1/1     Running   0
 4m
kube-prometheus-stack-prometheus-node-exporter-t9q7s     1/1     Running   0
 4m
kube-prometheus-stack-prometheus-node-exporter-x6vfb     1/1     Running   0
 4m
prometheus-kube-prometheus-stack-prometheus-0            2/2     Running   0
 4m
```

Lo stack distribuisce i seguenti componenti:

- Prometheus (StatefulSet): analizza le metriche e le scrive in remoto in AMP

- Grafana: dashboard e visualizzazione, preconfigurate con l'origine dati AMP
- kube-state-metrics: genera metriche sullo stato degli oggetti Kubernetes (stato del pod, risorsa, stati) requests/limits NodeClaim
- node-exporter (DaemonSet, una per nodo): raccoglie le metriche a livello di host (CPU, memoria, disco, rete)
- operatore: gestisce le risorse personalizzate Prometheus e Alertmanager

Alertmanager è disabilitato in questa configurazione.

Accedi a Grafana

È possibile accedere a Grafana tramite un AWS Application Load Balancer (ALB) connesso a Internet. Non lo crei qui. Il grafico kube-prometheus-stack lo fornisce dalla `grafana.ingress` configurazione nel file dei valori, utilizzando la sezione Set up load balancing. `alb IngressClass` ??? Il grafico Grafana Helm è dotato di supporto integrato, quindi non è necessario alcun manifest separato. `Ingress` L'ALB è denominato`grafana-ai-eks-docs`, riporta l'etichetta `guide: ai-eks-docs` ed è limitato all'indirizzo IP impostato `MY_CIDR` al momento dell'installazione.

> Il percorso di controllo dello stato di salute è obbligatorio per il gruppo target di Grafana ALB
>
> L'`alb.ingress.kubernetes.io/healthcheck-path`annotazione nel file dei valori indica i controlli di integrità del load balancer all'endpoint Grafana`/api/health`, che restituisce. `200` In caso contrario, i controlli di integrità ALB vengono impostati come predefiniti`/`, a cui Grafana risponde con un `302` reindirizzamento`/login`, facendo sì che il gruppo target segnali l'endpoint come non integro.

Stampa l'URL del load balancer. Il load balancer viene creato in modo asincrono, quindi attendi uno o due minuti:

```
echo "http://$(kubectl get ingress kube-prometheus-stack-grafana -n monitoring -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')"
```

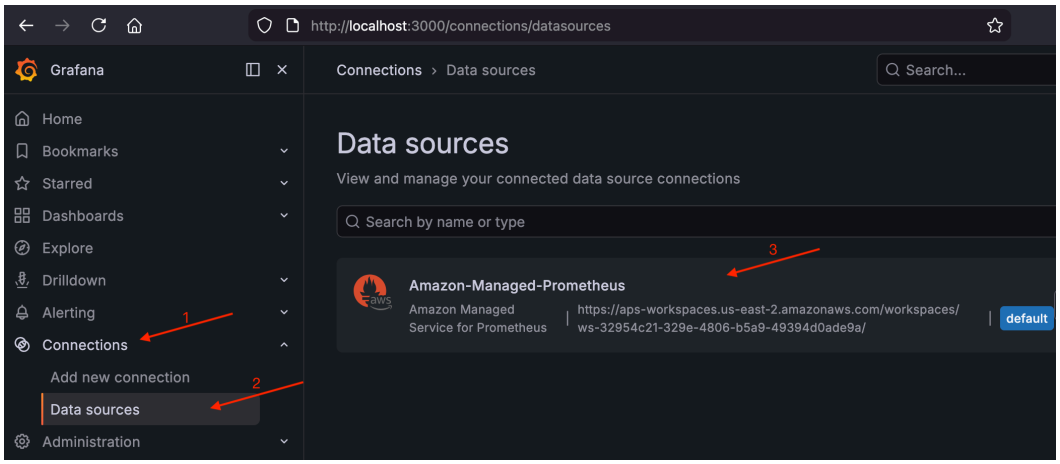
Apri l'URL nel tuo browser. Accedi con nome utente `admin` e password con il seguente comando:

```
kubectl --namespace monitoring get secrets kube-prometheus-stack-grafana -o
 jsonpath="{.data.admin-password}" | base64 -d ; echo
```

Per verificare che la pipeline delle metriche funzioni dall'inizio alla fine:

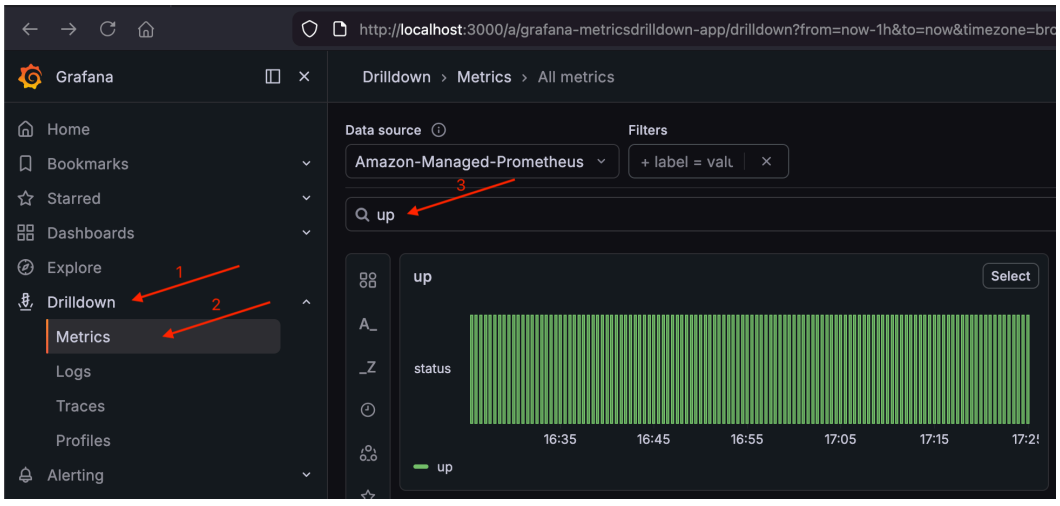
1. Vai a Connessioni > Origini dati e conferma `Amazon-Managed-Prometheus` che sia elencato come origine dati predefinita.

   Convalida l'origine dati AMP in Grafana




2. Vai a Drilldown > Metriche e cerca la metrica. up Dovresti vedere i risultati degli obiettivi di scrape del tuo cluster.

   Convalida la **up** metrica in Grafana




Se up vengono visualizzati i risultati, la pipeline (cluster → Prometheus → AMP → Grafana) funziona.

Implementa le metriche DCGM Exporter for GPU

Lo stack kube-prometheus-stack raccoglie le metriche di CPU e memoria a livello di nodo ma non le metriche della GPU. NVIDIA DCGM Exporter aggiunge l'utilizzo della GPU, l'utilizzo della memoria, la temperatura, il consumo energetico, la larghezza di banda NVLink e l'attività tensoriale.

```
helm repo add gpu-helm-charts https://nvidia.github.io/dcgm-exporter/helm-charts
helm repo update
```

Imposta la chiave di selezione del nodo GPU per il tuo percorso. EKS Auto Mode e Karpenter autogestito utilizzano chiavi di etichetta diverse per il produttore della GPU.

EKS Auto Mode

```
GPU_NODE_SELECTOR_KEY="eks.amazonaws.com/instance-gpu-manufacturer"
```

Self-managed Karpenter

```
GPU_NODE_SELECTOR_KEY="karpenter.k8s.aws/instance-gpu-manufacturer"
```

Crea il file dei valori dell'esportatore DCGM:

Example file di valori dcgm-exporter

```
cat << EOF > /tmp/dcgm-exporter-values.yaml
resources:
  requests:
    memory: "512Mi"
    cpu: "100m"
  limits:
    memory: "1Gi"
    cpu: "500m"

serviceMonitor:
  enabled: true
  additionalLabels:
    release: kube-prometheus-stack

nodeSelector:
  ${GPU_NODE_SELECTOR_KEY}: nvidia
```

```
tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"

customMetrics: |
  # Clocks
  DCGM_FI_DEV_SM_CLOCK,  gauge, SM clock frequency (in MHz).
  DCGM_FI_DEV_MEM_CLOCK, gauge, Memory clock frequency (in MHz).
  # Temperature
  DCGM_FI_DEV_MEMORY_TEMP, gauge, Memory temperature (in C).
  DCGM_FI_DEV_GPU_TEMP,    gauge, GPU temperature (in C).
  # Power
  DCGM_FI_DEV_POWER_USAGE,              gauge, Power draw (in W).
  DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION, counter, Total energy consumption since boot
 (in mJ).
  # PCIe
  DCGM_FI_PROF_PCIE_TX_BYTES,  counter, Number of bytes transmitted through PCIe TX (in
 KB) via NVML.
  DCGM_FI_PROF_PCIE_RX_BYTES,  counter, Number of bytes received through PCIe RX (in
 KB) via NVML.
  DCGM_FI_DEV_PCIE_REPLAY_COUNTER, counter, Total number of PCIe retries.
  # Utilization (the sample period varies depending on the product)
  DCGM_FI_DEV_GPU_UTIL,      gauge, GPU utilization (in %).
  DCGM_FI_DEV_MEM_COPY_UTIL, gauge, Memory utilization (in %).
  DCGM_FI_DEV_ENC_UTIL,      gauge, Encoder utilization (in %).
  DCGM_FI_DEV_DEC_UTIL,      gauge, Decoder utilization (in %).
  # Errors and violations
  DCGM_FI_DEV_XID_ERRORS,            gauge, Value of the last XID error encountered.
  DCGM_EXP_XID_ERRORS_COUNT,         gauge, Value of count of XID errors encountered.
  DCGM_FI_DEV_POWER_VIOLATION,       counter, Throttling duration due to power
 constraints (in us).
  DCGM_FI_DEV_THERMAL_VIOLATION,     counter, Throttling duration due to thermal
 constraints (in us).
  DCGM_FI_DEV_SYNC_BOOST_VIOLATION,  counter, Throttling duration due to sync-boost
 constraints (in us).
  DCGM_FI_DEV_BOARD_LIMIT_VIOLATION, counter, Throttling duration due to board limit
 constraints (in us).
  DCGM_FI_DEV_LOW_UTIL_VIOLATION,    counter, Throttling duration due to low
 utilization (in us).
  DCGM_FI_DEV_RELIABILITY_VIOLATION, counter, Throttling duration due to reliability
 constraints (in us).
  # Memory usage
  DCGM_FI_DEV_FB_FREE, gauge, Framebuffer memory free (in MiB).
```

```
 DCGM_FI_DEV_FB_USED, gauge, Framebuffer memory used (in MiB).
 # Retired pages
 DCGM_FI_DEV_RETIRED_SBE,     counter, Total number of retired pages due to single-bit
errors.
 DCGM_FI_DEV_RETIRED_DBE,     counter, Total number of retired pages due to double-bit
errors.
 DCGM_FI_DEV_RETIRED_PENDING, counter, Total number of pages pending retirement.
 # NVLink
 DCGM_FI_DEV_NVLINK_BANDWIDTH_TOTAL, counter, Total number of NVLink bandwidth
counters for all lanes.
 DCGM_FI_PROF_NVLINK_TX_BYTES,       counter, The rate of data transmitted over NVLink
not including protocol headers in bytes per second.
 DCGM_FI_PROF_NVLINK_RX_BYTES,       counter, The rate of data received over NVLink
not including protocol headers in bytes per second.
 # DCP metrics
 DCGM_FI_PROF_GR_ENGINE_ACTIVE,   gauge, Ratio of time the graphics engine is active
(in %).
 DCGM_FI_PROF_SM_ACTIVE,          gauge, The ratio of cycles an SM has at least 1 warp
assigned (in %).
 DCGM_FI_PROF_SM_OCCUPANCY,       gauge, The ratio of number of warps resident on an
SM (in %).
 DCGM_FI_PROF_PIPE_TENSOR_ACTIVE, gauge, Ratio of cycles the tensor (HMMA) pipe is
active (in %).
 DCGM_FI_PROF_DRAM_ACTIVE,        gauge, Ratio of cycles the device memory interface
is active sending or receiving data (in %).
 DCGM_FI_DEV_CLOCK_THROTTLE_REASONS, gauge, Current clock throttle reasons (bitmask of
DCGM_CLOCKS_THROTTLE_REASON_*).
 DCGM_FI_DEV_GPU_NVLINK_ERRORS,      gauge, Identifies a GPU NVLink error type
returned by DCGM_FI_DEV_GPU_NVLINK_ERRORS.
 ## NVLink
 DCGM_FI_DEV_NVLINK_BANDWIDTH_L0, counter, The number of bytes of active NVLink rx or
tx data including both header and payload.
 ## Remapped rows
 DCGM_FI_DEV_UNCORRECTABLE_REMAPPED_ROWS, counter, Number of remapped rows for
uncorrectable errors.
 DCGM_FI_DEV_CORRECTABLE_REMAPPED_ROWS, counter, Number of remapped rows for
correctable errors.
 DCGM_FI_DEV_ROW_REMAP_FAILURE, gauge, whether remapping of rows has failed.
 ## Profiling metrics
 DCGM_FI_PROF_PIPE_FP64_ACTIVE, gauge, Ratio of cycles the fp64 pipes are active (in
%).
 DCGM_FI_PROF_PIPE_FP32_ACTIVE, gauge, Ratio of cycles the fp32 pipes are active (in
%).
```

```
  DCGM_FI_PROF_PIPE_FP16_ACTIVE, gauge, Ratio of cycles the fp16 pipes are active (in
 %).
  # ECC
  DCGM_FI_DEV_ECC_SBE_VOL_TOTAL, counter, Total number of single-bit volatile ECC
 errors.
  DCGM_FI_DEV_ECC_DBE_VOL_TOTAL, counter, Total number of double-bit volatile ECC
 errors.
EOF
```

Il `customMetrics` campo sostituisce il set di metriche predefinito dell'esportatore DCGM con uno esteso che include larghezza di banda NVLink, attività tensoriale, throughput PCIe, errori ECC e limitazione termica. Per i carichi di lavoro di inferenza, aiutano a capire se le unità di elaborazione GPU sono completamente utilizzate, se la GPU è inattiva tra una richiesta e l'altra a causa delle dimensioni ridotte dei batch, se il trasferimento di dati tra CPU e GPU è un collo di bottiglia, se la limitazione termica sta causando picchi di latenza e quanto spazio di memoria della GPU rimane per batch più grandi.

Installa l'esportatore DCGM:

```
helm install dcgm-exporter gpu-helm-charts/dcgm-exporter \
  --namespace monitoring \
  -f /tmp/dcgm-exporter-values.yaml
```

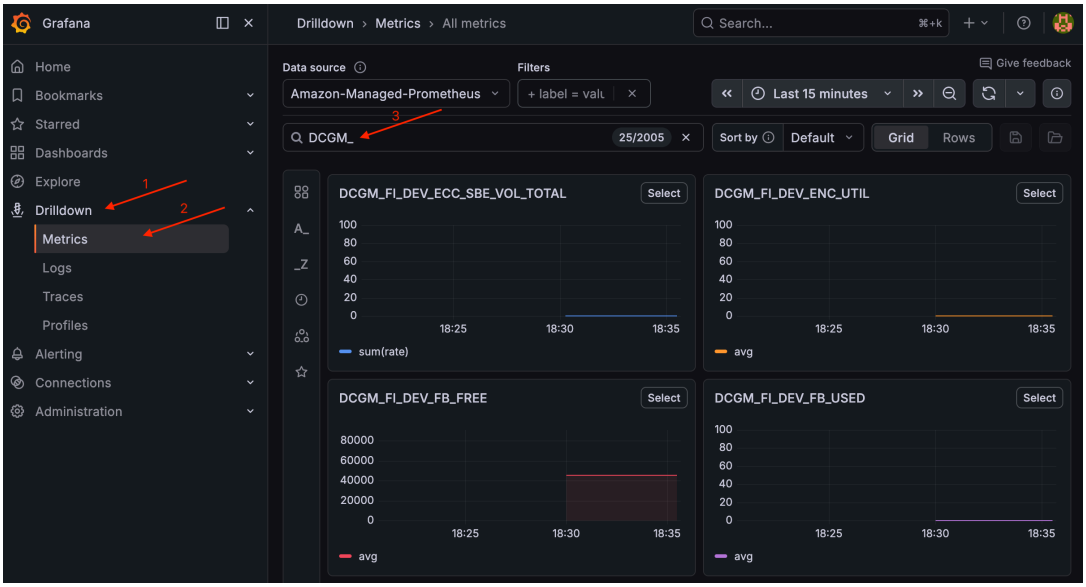
`tolerations`Consenti all'esportatore di funzionare sui GPU-tainted nodi di cui hai effettuato il provisioning nel passaggio 2. L'`serviceMonitor`release: `kube-prometheus-stack`etichetta assicura che Prometheus lo scopra e lo scarti automaticamente.

Verifica l'esportatore DCGM: DaemonSet

```
kubectl get daemonset dcgm-exporter -n monitoring
```

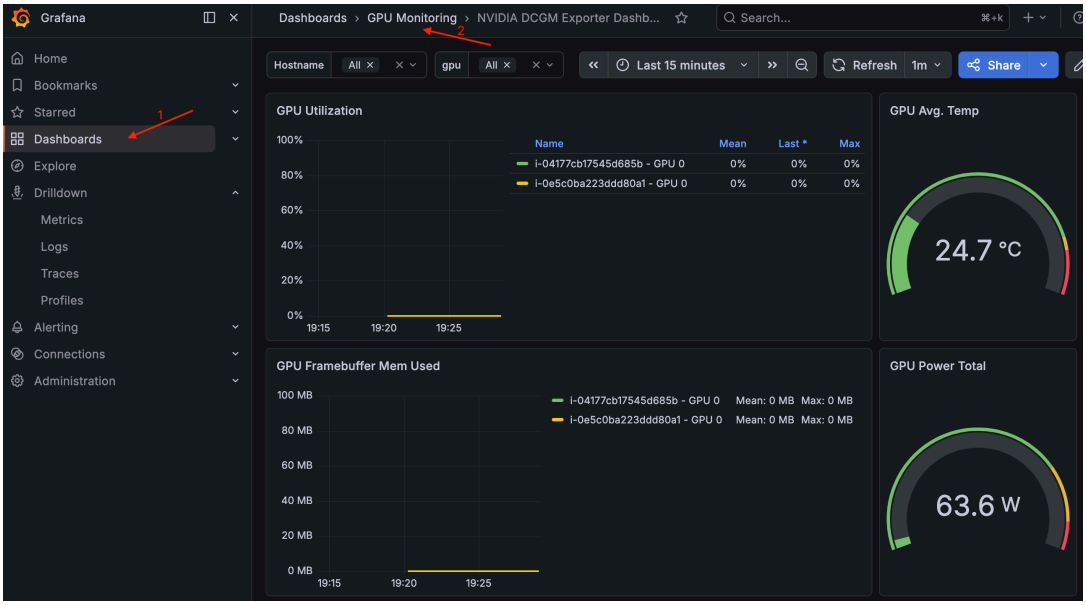
Una volta che un nodo GPU è in esecuzione, dovresti vedere un Pod pronto. Per convalidare le metriche DCGM, accedi a  Drilldown > Metrics in Grafana e cerca.  `DCGM_`

Convalida le metriche DCGM in Grafana




Per visualizzare la dashboard, vai su  Dashboard > Monitoraggio GPU > NVIDIA DCGM Exporter Dashboard.

Dashboard di NVIDIA DCGM Exporter a Grafana




## Il modello pesa un secchio S3

Crea un bucket Amazon S3 per memorizzare i pesi dei modelli e configura una EKS Pod Identity Association in modo che i workload pod possano leggervi e scrivere.

Se hai aperto un nuovo terminale, imposta il nome e la regione del cluster:

```
export CLUSTER_NAME=ai-eks-docs
export AWS_REGION=us-east-2
```

Crea il bucket S3

Crea il bucket con un suffisso casuale per evitare conflitti di nomi:

```
BUCKET_SUFFIX=$(head -c 4 /dev/urandom | od -An -tx1 | tr -d ' \n')
MODEL_BUCKET="${CLUSTER_NAME}-models-${BUCKET_SUFFIX}"

aws s3 mb s3://${MODEL_BUCKET} --region ${AWS_REGION}
```

I bucket S3 creati dopo gennaio 2023 hanno la crittografia lato server (AES256) e il blocco dell'accesso pubblico abilitati per impostazione predefinita.

Configura EKS Pod Identity per l'accesso a S3

Crea uno `model-storage-sa` ServiceAccount spazio dei `default` nomi, una policy IAM con ambito al bucket del modello e un'EKS Pod Identity Association che li colleghi. I workload pod impostati `serviceAccountName: model-storage-sa` saranno in grado di leggere e scrivere nel bucket.

```
kubectl create serviceaccount model-storage-sa
```

Crea la policy IAM:

```
POLICY_ARN=$(aws iam create-policy \
  --policy-name "${CLUSTER_NAME}-model-storage-policy" \
  --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\":
 \"Allow\", \"Action\": [\"s3:GetObject\", \"s3:PutObject\", \"s3:ListBucket\",
 \"s3:DeleteObject\"], \"Resource\": [\"arn:aws:s3:::${MODEL_BUCKET}\", \"arn:aws:s3:::
${MODEL_BUCKET}/*\"]}]}" \
  --query 'Policy.Arn' \
  --output text)

echo "Policy ARN: ${POLICY_ARN}"
```

> **Note**
>
> Questa policy garantisce `s3:DeleteObject` e `s3:PutObject` per la fase di convalida. Per i pod di inferenza di produzione che leggono solo i pesi dei modelli, `s3:PutObject` rimuovili e segui il privilegio minimo. `s3:DeleteObject`

Crea l'EKS Pod Identity Association. `eksctl`crea il ruolo IAM con la corretta policy di fiducia e lo collega a ServiceAccount:

```
eksctl create podidentityassociation \
  --cluster ${CLUSTER_NAME} \
  --namespace default \
  --service-account-name model-storage-sa \
  --role-name "${CLUSTER_NAME}-model-storage-role" \
  --permission-policy-arns ${POLICY_ARN} \
  --region ${AWS_REGION}
```

Verifica l'associazione:

```
eksctl get podidentityassociation --cluster ${CLUSTER_NAME} --region ${AWS_REGION}
```

L'output deve includere l'`model-storage-sa`associazione nel `default` namespace.

Convalida l'accesso S3 da un Pod

Esegui un Pod singolo con l'immagine AWS CLI, utilizzando `model-storage-sa` ServiceAccount, per confermare che EKS Pod Identity sia cablato e che l'accesso S3 funzioni:

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: s3-test
  labels:
    guide: ai-eks-docs
spec:
  serviceAccountName: model-storage-sa
  containers:
    - name: aws-cli
```

```
      image: public.ecr.aws/aws-cli/aws-cli:2.27.0
      command:
        - sh
        - -c
        - |
          echo "=== Caller Identity ==="
          aws sts get-caller-identity
          echo ""
          echo "=== S3 Write Test ==="
          echo "pod identity works" | aws s3 cp - s3://${MODEL_BUCKET}/test.txt
          echo ""
          echo "=== S3 List Test ==="
          aws s3 ls s3://${MODEL_BUCKET}/
          echo ""
          echo "=== S3 Delete Test ==="
          aws s3 rm s3://${MODEL_BUCKET}/test.txt
  restartPolicy: Never
EOF
```

Attendi il completamento del Pod e controlla i log:

```
kubectl wait --for=jsonpath='{.status.phase}'=Succeeded pod/s3-test --timeout=300s
kubectl logs s3-test
```

Output previsto:

```
=== Caller Identity ===
{
    "UserId": "AROA...:eks-ai-eks-docs-model-s-...",
    "Account": "123456789012",
    "Arn": "arn:aws:sts::123456789012:assumed-role/ai-eks-docs-model-storage-role/eks-
ai-eks-docs-model-s-..."
}

=== S3 Write Test ===
upload: - to s3://ai-eks-docs-models-01234567/test.txt

=== S3 List Test ===
2026-05-04 12:00:00         19 test.txt

=== S3 Delete Test ===
delete: s3://ai-eks-docs-models-01234567/test.txt
```

L'identità del chiamante conferma che il Pod ha assunto il `${CLUSTER_NAME}-model-storage-role` ruolo tramite EKS Pod Identity. I comandi S3 confermano l'accesso in lettura e scrittura.

Pulisci il pod di prova:

```
kubectl delete pod s3-test
```

## Fasi successive

Con il cluster pronto, puoi passare a Load & Serve Model per implementare un modello linguistico di grandi dimensioni e interagire con l'endpoint di inferenza.

## Pulizia

> **Tip**
>
> Se intendi continuare con le sezioni successive di questa guida, salta la pulizia completa. Eseguila solo quando hai finito.

```
export CLUSTER_NAME=ai-eks-docs
export AWS_REGION=us-east-2
```

```
kubectl delete pod nvidia-smi --ignore-not-found
kubectl delete deployment gpu-overflow-test --ignore-not-found
```

### Annulla la prenotazione della capacità

Se hai creato un ODCR, cancellalo prima. Cerca l'ID della prenotazione:

```
INSTANCE_TYPE="g6e.4xlarge"
CAPACITY_RESERVATION_ID=$(aws ec2 describe-capacity-reservations \
  --filters "Name=state,Values=active" "Name=instance-type,Values=${INSTANCE_TYPE}" \
  --query 'CapacityReservations[0].CapacityReservationId' \
  --output text \
  --region ${AWS_REGION})
echo "Capacity reservation ID: ${CAPACITY_RESERVATION_ID}"
```

Cancella la prenotazione:

```
aws ec2 cancel-capacity-reservation --capacity-reservation-id
 ${CAPACITY_RESERVATION_ID}
```

> **Important**
>
> L'annullamento di una prenotazione non interrompe le istanze in esecuzione. Continuano a essere applicate le On-Demand tariffe standard fino al termine. Elimina prima la distribuzione per svuotare il nodo riservato prima dell'annullamento.

Pulisci il monitoraggio

Cerca l'ARN della policy IAM:

```
AMP_POLICY_ARN=$(aws iam list-policies \
  --scope Local \
  --query "Policies[?PolicyName=='${CLUSTER_NAME}-amp-grafana-policy'].Arn" \
  --output text)
echo "AMP Policy ARN: ${AMP_POLICY_ARN}"
```

Cerca l'ID dello spazio di lavoro AMP:

```
AMP_WORKSPACE_ID=$(aws amp list-workspaces \
  --alias "amp-ws-${CLUSTER_NAME}" \
  --query 'workspaces[0].workspaceId' \
  --output text \
  --region ${AWS_REGION})
echo "AMP Workspace ID: ${AMP_WORKSPACE_ID}"
```

Disinstalla la versione Helm dell'esportatore DCGM:

```
helm uninstall dcgm-exporter -n monitoring
```

Disinstallare la versione Kube-prometheus-stack Helm:

```
helm uninstall kube-prometheus-stack -n monitoring
```

Elimina l'associazione EKS Pod Identity per l'account del servizio di acquisizione Prometheus:

```
eksctl delete podidentityassociation \
```

```
  --cluster ${CLUSTER_NAME} \
  --namespace monitoring \
  --service-account-name amp-iamproxy-ingest-service-account \
  --region ${AWS_REGION}
```

Elimina l'associazione EKS Pod Identity per l'account del servizio Grafana:

```
eksctl delete podidentityassociation \
  --cluster ${CLUSTER_NAME} \
  --namespace monitoring \
  --service-account-name grafana-sa \
  --region ${AWS_REGION}
```

Elimina la policy IAM utilizzata da Prometheus e Grafana:

```
aws iam delete-policy --policy-arn ${AMP_POLICY_ARN}
```

Elimina lo spazio di lavoro AMP:

```
aws amp delete-workspace --workspace-id ${AMP_WORKSPACE_ID} --region ${AWS_REGION}
```

Elimina lo spazio dei nomi di monitoraggio:

```
kubectl delete namespace monitoring
```

Pulisci i pesi del modello (secchio S3)

Cerca il nome del bucket del modello:

```
MODEL_BUCKET=$(aws s3api list-buckets \
  --query "Buckets[?starts_with(Name, '${CLUSTER_NAME}-models-')].Name | [0]" \
  --output text)
echo "Model bucket: ${MODEL_BUCKET}"
```

Cerca l'ARN della policy IAM:

```
POLICY_ARN=$(aws iam list-policies \
  --scope Local \
  --query "Policies[?PolicyName=='${CLUSTER_NAME}-model-storage-policy'].Arn" \
  --output text)
```

```
echo "Policy ARN: ${POLICY_ARN}"
```

Elimina il bucket del modello S3 e tutti i suoi oggetti:

```
aws s3 rb s3://${MODEL_BUCKET} --force
```

Elimina l'associazione EKS Pod Identity:

```
eksctl delete podidentityassociation \
  --cluster ${CLUSTER_NAME} \
  --namespace default \
  --service-account-name model-storage-sa \
  --region ${AWS_REGION}
```

Elimina la policy IAM:

```
aws iam delete-policy --policy-arn ${POLICY_ARN}
```

Elimina Kubernetes: ServiceAccount

```
kubectl delete serviceaccount model-storage-sa
```

Elimina le risorse rimanenti e il cluster

```
kubectl delete nodepool gpu-inf --ignore-not-found
kubectl delete nodeclass gpu-inf --ignore-not-found
kubectl delete ec2nodeclass gpu-inf --ignore-not-found
eksctl delete cluster --name=$CLUSTER_NAME --region=$AWS_REGION
```

## Configura il cluster Amazon EKS per i AI/ML carichi di lavoro utilizzando Terraform

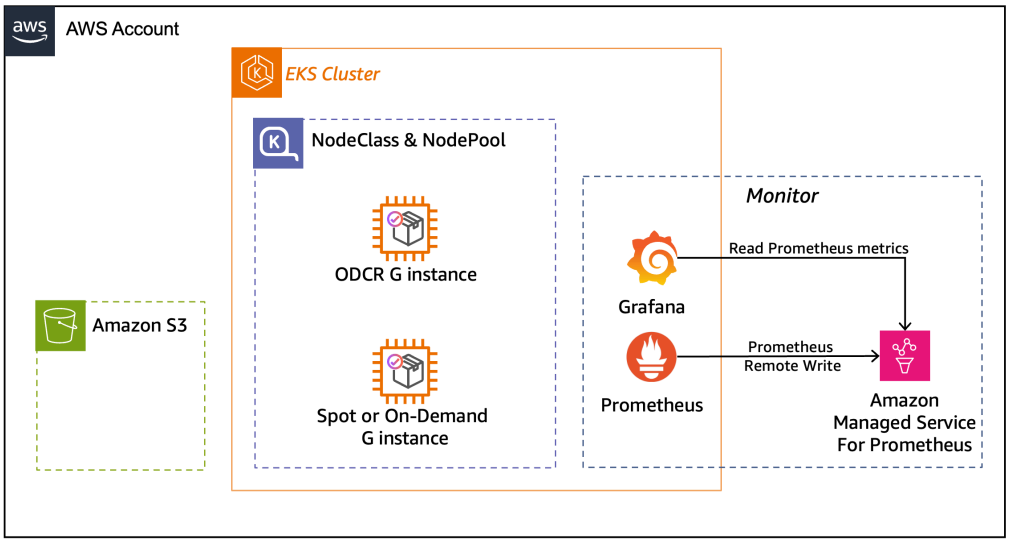
> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS. AI/ML

Questa sezione illustra i passaggi per creare l'infrastruttura necessaria per eseguire carichi di lavoro di formazione o inferenza su Amazon EKS utilizzando Terraform. I passaggi includono la creazione di

un cluster EKS, GPU-enabled nodi con EKS Auto Mode o Karpenter, uno stack di monitoraggio con Prometheus e Grafana e lo storage Amazon S3 per i pesi dei modelli.

Consulta la documentazione per EKS Auto Mode e Karpenter per ulteriori informazioni su come queste funzionalità forniscono e scalano automaticamente le istanze EC2 nei cluster EKS.

High-level architettura e flusso di lavoro




Il diagramma mostra l'architettura AWS di alto livello per la configurazione di questa sezione.

## Prerequisiti

> **Important**
>
> Le risorse create in questo tutorial, tra cui cluster EKS, istanze GPU, Application Load Balancer e Amazon Managed Service for Prometheus, sono a pagamento. Elimina le risorse quando hai finito per evitare addebiti continui.

- Terraform >= 1.15.0. Per le istruzioni di configurazione, vedi Installazione di Terraform. https://developer.hashicorp.com/terraform/install
- `kubectl`>= 1.36. Per le istruzioni di configurazione, vedere. the section called "Configura kubectl ed eksctl"
- AWS CLI >= 2.27. Per le istruzioni di configurazione, vedere Installazione. https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html

- `jq`. Per le istruzioni di configurazione, consulta Download jq.

Verifica le versioni dei tuoi strumenti:

```
terraform --version
aws --version
kubectl version --client
jq --version
```

## Passaggio 1: scarica e distribuisci il codice Terraform

Questa procedura dettagliata utilizza il codice Terraform nel repository sample-eks-docs Samples. AWS GitHub Clona il repository in una directory di lavoro:

```
git clone git@github.com:aws-samples/sample-eks-docs.git
cd sample-eks-docs/ai-ml/set-up-cluster
```

Il repository ha la seguente struttura `ai-ml/set-up-cluster/` nella directory in cui hai appena cambiato:

```
set-up-cluster/
### scripts/
#   ### cleanup.sh
### terraform/
    ### auto-mode/
    ### karpenter/
```

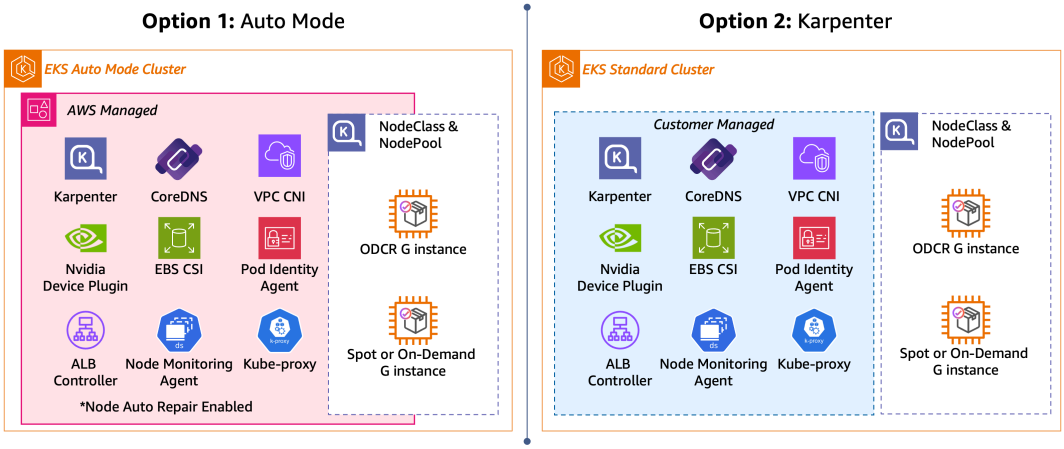
Il repository fornisce due percorsi di distribuzione. Scegline solo uno e utilizzalo in tutta la guida.

- EKS Auto Mode (`terraform/auto-mode/`) — Oltre ai principali componenti aggiuntivi di rete, storage e bilanciamento del carico, EKS Auto Mode include e gestisce le seguenti funzionalità per l'addestramento e l'inferenza dei carichi di lavoro: agente di monitoraggio dei nodi EKS, riparazione automatica dei nodi, snapshotter SOCI per una rapida estrazione dei container e predisposizione della GPU per l'impostazione predefinita. NodeClass Il plug-in per dispositivi NVIDIA è incluso nell'AMI accelerata Bottlerocket che EKS Auto Mode utilizza per i nodi. GPU-enabled
- Self-managed Karpenter (`terraform/karpenter/`) — Su un cluster EKS senza EKS Auto Mode, il codice Terraform installa e configura i componenti necessari per i carichi di lavoro di addestramento e inferenza. Ciò include componenti aggiuntivi di rete (VPC CNI, CoreDNS, kube-

proxy), Karpenter, l'agente di monitoraggio dei nodi EKS, il plug-in per dispositivi NVIDIA e lo snapshotter SOCI per richiamare rapidamente i container.

> **Important**
>
> Scegli EKS Auto Mode o Karpenter autogestito e usalo in tutta la guida. Il passaggio a metà flusso richiede la distruzione del cluster e il riavvio da capo.

Opzioni del cluster EKS: EKS Auto Mode e Karpenter autogestito




> **Grafana è accessibile pubblicamente tramite HTTP con credenziali predefinite**
>
> L'`Ingress`impostazione predefinita di Grafana ALB è`0.0.0.0/0`, che espone Grafana `var.my_cidr` alla rete Internet pubblica tramite un semplice HTTP con credenziali di amministratore predefinite. Gli scanner automatici scoprono i sistemi di bilanciamento del carico pubblici in pochi minuti. È necessario limitare l'accesso sovrascrivendo il proprio `var.my_cidr` indirizzo IP:
>
> ```
> export MY_CIDR="$(curl -s https://checkip.amazonaws.com)/32"
> terraform apply -var "my_cidr=${MY_CIDR}"
> ```
>
> Considera l'elenco degli indirizzi IP di origine come una protezione minima, non completa. Cambia anche la password amministratore predefinita di Grafana dopo il primo accesso. Per una postura più sicura, modifica l'`alb.ingress.kubernetes.io/scheme`impostazione

`internal` (raggiungibile solo dall'interno del tuo VPC o da una VPN connessa) e aggiungi un certificato TLS.

Distribuzione del cluster

Entra nella directory del percorso scelto, inizializza Terraform e applica:

Entrambe le varianti utilizzano per impostazione predefinita la regione. `us-east-2` Per eseguire la distribuzione in una regione diversa, aggiungi `-var "region=`*`region-code`*`"` al `terraform apply` comando nel passaggio successivo, *`region-code`* dov'è la AWS regione in cui desideri eseguire la distribuzione.

Il codice Terraform utilizza tutte le zone di disponibilità disponibili nella regione di destinazione `use1-az3usw1-az2`, escluse e `cac1-az3` perché Amazon EKS non supporta il posizionamento dei piani di controllo in quelle zone.

EKS Auto Mode

```
cd terraform/auto-mode
terraform init
terraform apply
```

Il completamento di questo comando richiede alcuni minuti.

Self-managed Karpenter

```
cd terraform/karpenter
terraform init
terraform apply
```

Questo comando richiede circa 15 minuti. Crea un cluster EKS con un gruppo di nodi gestiti dedicato all'hosting di componenti aggiuntivi e il controller Karpenter. Terraform installa Karpenter con la coda di interruzione Spot abilitata e i gate delle funzionalità e attivati. `NodeRepair` `StaticCapacity` Installa anche il plug-in del dispositivo NVIDIA, il Load Balancer Controller e lo AWS stack di monitoraggio.

### Esamina gli output di Terraform

Al termine dell'applicazione, Terraform stampa i seguenti output (i valori variano in base alla configurazione):

```
Apply complete! Resources: 74 added, 0 changed, 0 destroyed.

Outputs:

cluster_name         = "ai-eks-docs"
configure_kubectl    = "aws eks update-kubeconfig --region us-east-2 --name ai-eks-docs
 --alias ai-eks-docs"
configure_model_bucket = "export MODEL_BUCKET=ai-eks-docs-models-20250612abc1"
model_bucket         = "ai-eks-docs-models-20250612abc1"
node_iam_role_name   = "ai-eks-docs-eks-auto-20250612..."
region               = "us-east-2"
```

L'`configure_kubectl`output è un comando pronto per l'esecuzione che punta al cluster. `kubectl` L'`model_bucket`output contiene il nome del bucket S3 per i pesi del modello. L'`node_iam_role_name`output mostra il ruolo IAM utilizzato dai nodi.

### Configura kubectl

Indicate il `kubectl` nuovo cluster. L'`configure_kubectl`output è un comando pronto per l'esecuzione:

```
eval "$(terraform output -raw configure_kubectl)"
```

### Verifica il cluster

#### EKS Auto Mode

```
kubectl get pods --all-namespaces
```

Output previsto:

```
NAMESPACE     NAME                                                  READY
 STATUS    RESTARTS   AGE
kube-system   metrics-server-5db89f9ffd-h4mlr                       1/1
 Running   0          3m
kube-system   metrics-server-5db89f9ffd-nd748                       1/1
 Running   0          3m
```

```
monitoring    kube-prometheus-stack-grafana-ff9b5fd57-dh562              3/3
 Running   0          3m
monitoring    kube-prometheus-stack-kube-state-metrics-5dcbfdf69b-wd6qn  1/1
 Running   0          3m
monitoring    kube-prometheus-stack-operator-548c4f4485-m5wh2            1/1
 Running   0          3m
monitoring    kube-prometheus-stack-prometheus-node-exporter-p2z7l       1/1
 Running   0          3m
monitoring    prometheus-kube-prometheus-stack-prometheus-0              2/2
 Running   0          3m
```

In EKS Auto Mode, VPC CNI, kube-proxy e CoreDNS vengono eseguiti come componenti gestiti e non vengono visualizzati come pod in. `kube-system`

Self-managed Karpenter

```
kubectl get pods --all-namespaces
```

L'output previsto include Karpenter, CoreDNS, kube-proxy, aws-node (VPC CNI), EKS Pod Identity Agent, l'agente di monitoraggio dei nodi EKS e il plug-in per i dispositivi NVIDIA:

```
NAMESPACE     NAME
 READY   STATUS    RESTARTS   AGE
kube-system   aws-node-bzdcz                                             2/2
    Running   0          5m
kube-system   aws-node-vkbhb                                             2/2
    Running   0          5m
kube-system   coredns-7dbb8998cf-9b9wk                                   1/1
    Running   0          5m
kube-system   coredns-7dbb8998cf-pwtjd                                   1/1
    Running   0          5m
kube-system   ebs-csi-controller-748f54b69-8h7jv                         6/6
    Running   0          5m
kube-system   ebs-csi-controller-748f54b69-v2mv8                         6/6
    Running   0          5m
kube-system   eks-node-monitoring-agent-5qw4l                            1/1
    Running   0          5m
kube-system   eks-node-monitoring-agent-7lrtm                            1/1
    Running   0          5m
kube-system   eks-pod-identity-agent-ddvlv                               1/1
    Running   0          5m
kube-system   eks-pod-identity-agent-q4g29                               1/1
    Running   0          5m
```

```
kube-system   karpenter-898ff78-cndbd                                          1/1
    Running   0          5m
kube-system   karpenter-898ff78-hfbhn                                          1/1
    Running   0          5m
kube-system   kube-proxy-gcfnh                                                 1/1
    Running   0          5m
kube-system   kube-proxy-tktcf                                                 1/1
    Running   0          5m
kube-system   metrics-server-5b789db597-cm9qd                                   1/1
    Running   0          5m
kube-system   metrics-server-5b789db597-w57b6                                   1/1
    Running   0          5m
kube-system   nvidia-device-plugin-node-feature-discovery-gc-66cb7f5dc-xvftc    1/1
    Running   0          5m
kube-system   nvidia-device-plugin-node-feature-discovery-master-854d6b5hnqtp   1/1
    Running   0          5m
monitoring    kube-prometheus-stack-grafana-6797bcb59f-2zlwg                   3/3
    Running   0          5m
monitoring    kube-prometheus-stack-kube-state-metrics-8446bd549c-q6gjc        1/1
    Running   0          5m
monitoring    kube-prometheus-stack-operator-5f499b784-vf5xb                   1/1
    Running   0          5m
monitoring    kube-prometheus-stack-prometheus-node-exporter-4c6wq             1/1
    Running   0          5m
monitoring    kube-prometheus-stack-prometheus-node-exporter-l5t25             1/1
    Running   0          5m
monitoring    prometheus-kube-prometheus-stack-prometheus-0                    2/2
    Running   0          5m
```

Verifica che il plug-in del dispositivo NVIDIA sia installato. Zero Device Plugin Pod viene visualizzato finché non viene fornita una GPU NodePool con l'`amiFamily=al2023`etichetta corrispondente:

```
kubectl get daemonset nvidia-device-plugin -n kube-system
```

Output previsto:

```
NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR      AGE
nvidia-device-plugin   0         0         0       0            0
 amiFamily=al2023   5m
```

## Passaggio 2: crea una GPU dinamica NodePool

Le GPU NodePools sono attivate. Per impostazione predefinita, `terraform apply` crea il cluster e lo stack di monitoraggio senza capacità GPU e senza fatturazione GPU. Per eseguire il provisioning dei nodi GPU, passate la `nodepools` variabile con un nome di strategia.

Attivate la `spot-ondemand` strategia, che prevede il provisioning di istanze G-family GPU con una generazione superiore a 4, utilizzando la capacità Spot con On-Demand come riserva:

```
terraform apply -var 'nodepools={"spot-ondemand"={}}'
```

Questo comando applica i NodeClass modelli NodePool e contenuti nella directory. `nodepools/spot-ondemand/` Entrambi i percorsi utilizzano la stessa NodePool API, ma differiscono NodeClass nei NodePool riferimenti.

EKS Auto Mode

Fanno NodePool riferimento a managed `default` NodeClass, che già seleziona l'AMI accelerata Bottlerocket, i driver NVIDIA, il plug-in del dispositivo NVIDIA e il pull parallelo SOCI. La `spot-ondemand` strategia non si prefigge di per sé questa strada. NodeClass

Convalida: NodePool

```
kubectl get nodepools,nodeclasses
```

Output previsto. I `gpu-inf` NodePool join incorporati e tutti `general-purpose` e `system` NodePools tre fanno riferimento al gestore: `default` NodeClass

```
NAME                                  NODECLASS   NODES   READY   AGE
nodepool.karpenter.sh/general-purpose   default     0       True    12m
nodepool.karpenter.sh/gpu-inf           default     0       True    20s
nodepool.karpenter.sh/system            default     1       True    12m

NAME                                ROLE                        READY   AGE
nodeclass.eks.amazonaws.com/default   ai-eks-docs-eks-auto-...   True    12m
```

Self-managed Karpenter

Terraform applica una personalizzazione `gpu-inf` `EC2NodeClass` insieme a. NodePool Inserisce `EC2NodeClass` l'alias AMI EKS-optimized AL2023, abilita SOCI attraverso il

`FastImagePull` feature gate e imposta lo spostamento della cache delle immagini contenuta `instanceStorePolicy: RAID0` su NVMe locali.

Convalida la e: NodePool EC2NodeClass

```
kubectl get nodepools,ec2nodeclasses
```

Output previsto. La `gpu-inf` coppia si unisce alla band creata da Terraform per carichi di lavoro non GPU: `general-purpose` NodePool EC2NodeClass

```
NAME                                    NODECLASS          NODES   READY   AGE
nodepool.karpenter.sh/general-purpose   general-purpose    1       True    14m
nodepool.karpenter.sh/gpu-inf           gpu-inf            0       True    25s

NAME                                                READY   AGE
ec2nodeclass.karpenter.k8s.aws/general-purpose      True    14m
ec2nodeclass.karpenter.k8s.aws/gpu-inf              True    25s
```

Entrambi i percorsi mostrano i `0` nodi `gpu-inf` fino alla pianificazione del carico di lavoro della GPU. EKS Auto Mode e Karpenter avviano i nodi solo quando i Pod in sospeso li richiedono.

## Fase 3: Esegui il test con un Pod di esempio

Verifica la NodePool configurazione della GPU con un `nvidia-smi` Pod:

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
  labels:
    guide: ai-eks-docs
spec:
  restartPolicy: OnFailure
  tolerations:
    - key: "nvidia.com/gpu"
      operator: "Exists"
      effect: "NoSchedule"
  containers:
    - name: nvidia-smi
```

```
      image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
      command: ["nvidia-smi"]
      resources:
        limits:
          nvidia.com/gpu: 1
EOF
```

Verifica che il Pod sia pianificato e completato correttamente:

```
kubectl get pods nvidia-smi
```

Output previsto:

```
NAME         READY   STATUS      RESTARTS   AGE
nvidia-smi   0/1     Completed   0          67s
```

Indica `STATUS: Completed` che il `nvidia-smi` comando è stato eseguito ed è terminato. Controlla i log del Pod per vedere la GPU rilevata dal nodo:

```
kubectl logs nvidia-smi
```

Output previsto:

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 580.159.03             Driver Version: 580.159.03     CUDA Version: 13.0
   |
+-----------------------------------------+------------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
 ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute
 M. |
|                                         |                        |               MIG
 M. |
|=========================================+========================
+======================|
|   0  NVIDIA L4                      On  |   00000000:31:00.0 Off |
 0 |
| N/A   41C    P8             13W /   72W |       0MiB /  23034MiB |      0%
 Default |
```

```
|                                         |                        |
 N/A |
+-----------------------------------------+------------------------
+----------------------+
```

L'output mostra il modello della GPU, la versione del driver, la versione CUDA e la memoria disponibile. In questo esempio, Karpenter ha effettuato il provisioning di un'istanza G6 con una GPU NVIDIA L4 con 24 GB di memoria. Il modello e la memoria della GPU variano a seconda del tipo di istanza selezionato da Karpenter. Le istanze G5 hanno GPU NVIDIA A10G (24 GB), le istanze G6 hanno GPU NVIDIA L4 (24 GB) e le istanze G6e hanno GPU NVIDIA L40S (48 GB).

Per capire come Karpenter e lo scheduler Kubernetes si sono coordinati per fornire un nodo e posizionare il Pod, controlla gli eventi del ciclo di vita del Pod:

```
kubectl describe pod nvidia-smi
```

Output previsto:

```
Events:
  Type     Reason            Age   From                   Message
  ----     ------            ----  ----                   -------
  Warning  FailedScheduling  75s   default-scheduler      0/2 nodes are available: 2
 node(s) had untolerated taint(s).
  Normal   Nominated         74s   eks-auto-mode/compute  Pod should schedule on:
 nodeclaim/gpu-inf-z6q75
  Normal   Scheduled         35s   default-scheduler      Successfully assigned
 default/nvidia-smi to i-0eb897a8302551589
  Normal   Pulling           27s   kubelet                spec.containers{nvidia-smi}:
 Pulling image "public.ecr.aws/amazonlinux/amazonlinux:2023-minimal"
  Normal   Pulled            22s   kubelet                spec.containers{nvidia-smi}:
 Successfully pulled image "public.ecr.aws/amazonlinux/amazonlinux:2023-minimal" in
 5.625s (5.626s including waiting). Image size: 37440620 bytes.
  Normal   Created           22s   kubelet                spec.containers{nvidia-smi}:
 Container created
  Normal   Started           21s   kubelet                spec.containers{nvidia-smi}:
 Container started
```

Questi eventi mostrano la sequenza di pianificazione del Pod: il Pod inizialmente non riesce a pianificare perché non esistono nodi GPU (`FailedScheduling`), Karpenter ne nomina un nuovo (), lo scheduler assegna il Pod una volta che il nodo è pronto NodeClaim (`Nominated`), quindi l'immagine del contenitore viene estratta e avviata. `Scheduled` EKS Auto Mode ha il pull parallelo

SOCI (Seekable OCI) installato e configurato di default sulle istanze G, P e Trn e il percorso Karpenter autogestito lo configura esplicitamente tramite il `FastImagePull` feature gate.

> **Note**
>
> In un cluster Karpenter autogestito, l'evento viene visualizzato al posto di. `Nominated` `karpenter/compute` `eks-auto-mode/compute`

A NodeClaim è una richiesta creata da Karpenter per eseguire il provisioning di un nodo specifico. Mostra il tipo di istanza, il tipo di capacità, l'AZ e se il nodo è pronto:

```
kubectl get nodeclaims
```

Output previsto:

```
NAME            TYPE        CAPACITY   ZONE         NODE                   READY   AGE
gpu-inf-z6q75   g6.xlarge   spot       us-east-2a   i-0eb897a8302551589   True    5m
```

Il tipo di istanza e l'AZ variano. Qualsiasi G-family istanza con una generazione superiore a 4 è idonea.

> **Tip**
>
> Se non viene visualizzato alcun nodo, verifica la presenza di errori di capacità insufficiente:
>
> ```
> kubectl get events | grep InsufficientCapacityError
> ```
>
> Karpenter memorizza nella cache le offerte non disponibili per 3 minuti. L'ampliamento dei tipi di istanze e degli AZ consentiti NodePool aumenta le possibilità di atterraggio.

> **Note**
>
> Le istanze Spot lanciate da Karpenter non vengono visualizzate nella console Spot Requests di EC2. Karpenter utilizza l'API EC2 `CreateFleet` con. `type: instant` Le istanze vengono visualizzate nella console EC2 Instances con un ciclo di vita. `spot`

## Fase 4: Aggiungere capacità riservata a (opzionale) NodePool

Sebbene la GPU della Fase 2 NodePool esegua il provisioning di Spot o On-Demand delle istanze in modo dinamico, alcuni casi d'uso richiedono una capacità garantita. Puoi creare un On-Demand Capacity Reservation (ODCR) per assicurarti che la capacità della GPU sia disponibile quando necessario.

Con Terraform, un singolo comando crea l'ODCR, una personalizzazione NodeClass che fa riferimento alla prenotazione per tag e lo aggiorna NodePool per includerlo `reserved` come tipo di capacità. Terraform contrassegna l'ODCR con `nodepool=reserved-spot-ondemand` e poi lo NodeClass seleziona in base a quel tag.

> **Warning**
>
> Il comando seguente crea un ODCR che viene fatturato immediatamente e continua a fatturare finché non viene distrutto con `terraform destroy` o con lo script di pulizia, indipendentemente dal fatto che i nodi siano in esecuzione o meno.

Usa i valori predefiniti (`g6e.4xlarge`, 1 istanza, primo cluster AZ):

```
terraform apply -var 'nodepools={"reserved-spot-ondemand"={reservation={}}}'
```

Scegli il tipo di istanza, il numero e la AZ:

```
terraform apply -var 'nodepools={"reserved-spot-
ondemand"={reservation={instance_type="g6e.2xlarge",instance_count=1,az="us-
east-2a"}}}'
```

L'`reservation`oggetto supporta i seguenti campi:

- `instance_type`— Il tipo di istanza GPU da prenotare. Default: `g6e.4xlarge`.
- `instance_count`— Il numero di istanze da prenotare. Default: `1`.
- `az`— La zona di disponibilità per la prenotazione. Predefinito: `""` (utilizza il primo cluster AZ).

> **Important**
>
> Le `reserved-spot-ondemand` strategie `spot-ondemand` e si escludono a vicenda. Puoi abilitarne al massimo una nella `nodepools` variabile. Se in precedenza lo si utilizzava `spot-ondemand` nel passaggio 2, il `reserved-spot-ondemand` comando lo sostituisce perché entrambi gestiscono la stessa cosa `gpu-inf` NodePool.

Se si `InsufficientInstanceCapacity` verifica un errore, la prenotazione non può essere soddisfatta nell'AZ specificato. Annulla l'operazione Terraform (Ctrl+C), quindi eseguila nuovamente con un valore diverso: `az`

```
terraform apply -var 'nodepools={"reserved-spot-
ondemand"={reservation={instance_type="g6e.4xlarge",az="us-east-2b"}}}'
```

Dopo l'applicazione, Terraform li aggiorna NodePool per include`reserved`, e nei requisiti relativi al tipo di `spot` capacità. `on-demand` Karpenter considera l'opzione più conveniente e `reserved` la lancia per prima. Una volta completata, la prenotazione torna a Spot o. On-Demand

Nel percorso EKS Auto Mode, Terraform crea un codice personalizzato `gpu-inf` NodeClass (perché il pacchetto `default` NodeClass è di sola lettura) che fa riferimento all'ODCR tramite tag. `capacityReservationSelectorTerms` Sul percorso Karpenter autogestito, Terraform riapplica le impostazioni aggiunte e le aggiorna per includere. `gpu-inf` EC2NodeClass `capacityReservationSelectorTerms` NodePool `reserved`

Verifica che l'ODCR sia stato creato:

```
aws ec2 describe-capacity-reservations \
  --filters "Name=state,Values=active" "Name=tag:nodepool,Values=reserved-spot-
ondemand" \
  --query 'CapacityReservations[0].
{Id:CapacityReservationId,State:State,InstanceType:InstanceType,AvailableCount:AvailableInstanc
 \
  --output table \
  --region $(terraform output -raw region)
```

Verifica i NodeClass riferimenti all'ODCR:

EKS Auto Mode

```
kubectl get nodeclasses gpu-inf -o yaml | grep 'id: cr-.*'
```

Output previsto:

```
        id: cr-xxxxxxxxxxxxxxxxx
```

Self-managed Karpenter

```
kubectl get ec2nodeclasses gpu-inf -o yaml | grep 'id: cr-.*'
```

Output previsto:

```
        id: cr-xxxxxxxxxxxxxxxxx
```

Verifica che NodePool sia pronto:

```
kubectl get nodepools gpu-inf
```

Output previsto:

```
NAME      NODECLASS   NODES   READY   AGE
gpu-inf   gpu-inf     0       True    30s
```

Verifica che la capacità riservata venga utilizzata per prima, con il fallback di Spot

Dopo aver applicato le modifiche, verifica che Karpenter dia la priorità alla capacità riservata e ritorni a Spot o. On-Demand Implementa una distribuzione con 2 repliche che richiede 1 GPU per pod. L'ODCR è per 1 istanza (1 GPU), quindi il primo Pod attiva Karpenter per avviare un nodo riservato. Il secondo Pod non può essere inserito nel nodo riservato e fa sì che Karpenter avvii un altro nodo da Spot o Capacity. On-Demand

```
cat << 'EOF' | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpu-overflow-test
```

```
  labels:
    guide: ai-eks-docs
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gpu-overflow-test
  template:
    metadata:
      labels:
        app: gpu-overflow-test
        guide: ai-eks-docs
    spec:
      tolerations:
        - key: nvidia.com/gpu
          operator: Exists
          effect: NoSchedule
      containers:
        - name: nvidia-smi
          image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
          command: ["sh", "-c", "nvidia-smi && sleep infinity"]
          resources:
            limits:
              nvidia.com/gpu: 1
EOF
```

A differenza del Pod di `nvidia-smi` test di Step 3, che è stato eseguito e chiuso, questo Deployment mantiene i Pod in funzione (`sleep infinity`) in modo da trattenere la GPU e impedire il consolidamento del nodo.

Verifica i Pod pianificati su diversi nodi:

```
kubectl get pods -l app=gpu-overflow-test -o wide
```

Output previsto:

```
NAME                                READY   STATUS    RESTARTS   AGE     IP
 NODE                  NOMINATED NODE   READINESS GATES
gpu-overflow-test-55d55ff5b9-dvg52   1/1     Running   0          4m42s   10.0.75.210
 i-08741a36089ff2088   <none>           <none>
gpu-overflow-test-55d55ff5b9-hw4m9   1/1     Running   0          4m43s   10.0.82.49
 i-0f50cdbacb2017202   <none>           <none>
```

Controlla NodeClaims per vedere i tipi di capacità:

```
kubectl get nodeclaims
```

Output previsto:

```
NAME            TYPE          CAPACITY   ZONE         NODE                   READY   AGE
gpu-inf-vw99m   g6e.4xlarge   reserved   us-east-2c   i-0f50cdbacb2017202   True    6m
gpu-inf-s65s6   g6.xlarge     spot       us-east-2b   i-08741a36089ff2088   True
 5m59s
```

Il nodo riservato viene avviato per primo, seguito da uno Spot o da un On-Demand nodo una volta completata la prenotazione.

Pulisci la distribuzione di prova:

```
kubectl delete deployment gpu-overflow-test
```

## Monitoraggio

Terraform ha già fornito l'intero stack di monitoraggio durante la fase 1`terraform apply`. Lo stack include uno spazio di lavoro Amazon Managed Service for Prometheus (AMP), policy IAM e EKS Pod Identity Associations per l'accesso alla scrittura remota e alle query Grafana di Prometheus, il grafico Helm kube-prometheus-stack (Prometheus, Grafana, kube-state-metrics, node-exporter) e NVIDIA DCGM Exporter per le metriche GPU.

Questa sezione riguarda la verifica dei componenti di monitoraggio implementati.

Verifica i pod di monitoraggio

Attendi che tutti i pod di monitoraggio siano pronti:

```
kubectl wait --for=condition=Ready pod --all -n monitoring --timeout=300s
kubectl get pods -n monitoring
```

Output previsto:

```
NAME                                                     READY   STATUS    RESTARTS
 AGE
```

```
kube-prometheus-stack-grafana-7c58f54f77-rftrj             3/3     Running   0
 5m
kube-prometheus-stack-kube-state-metrics-d68dcbc84-5smxq   1/1     Running   0
 5m
kube-prometheus-stack-operator-5895df479f-ttm47            1/1     Running   0
 5m
kube-prometheus-stack-prometheus-node-exporter-t9q7s       1/1     Running   0
 5m
kube-prometheus-stack-prometheus-node-exporter-x6vfb       1/1     Running   0
 5m
prometheus-kube-prometheus-stack-prometheus-0              2/2     Running   0
 5m
```

Accedi a Grafana

Grafana è esposta tramite un AWS Application Load Balancer (ALB) connesso a Internet, limitato al CIDR impostato. `var.my_cidr` Stampa l'URL del load balancer (attendi uno o due minuti per il provisioning dell'ALB):

```
echo "http://$(kubectl get ingress kube-prometheus-stack-grafana -n monitoring -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')"
```

Apri l'URL nel tuo browser. Accedi con nome utente `admin` e password con il seguente comando:

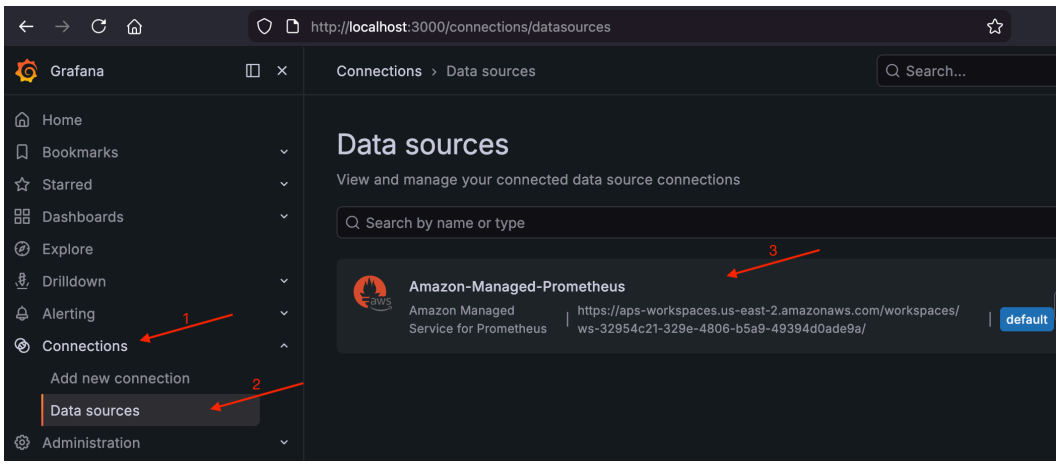
```
kubectl --namespace monitoring get secrets kube-prometheus-stack-grafana -o
 jsonpath="{.data.admin-password}" | base64 -d ; echo
```

Verifica la pipeline delle metriche

Per verificare che la pipeline delle metriche funzioni integralmente:

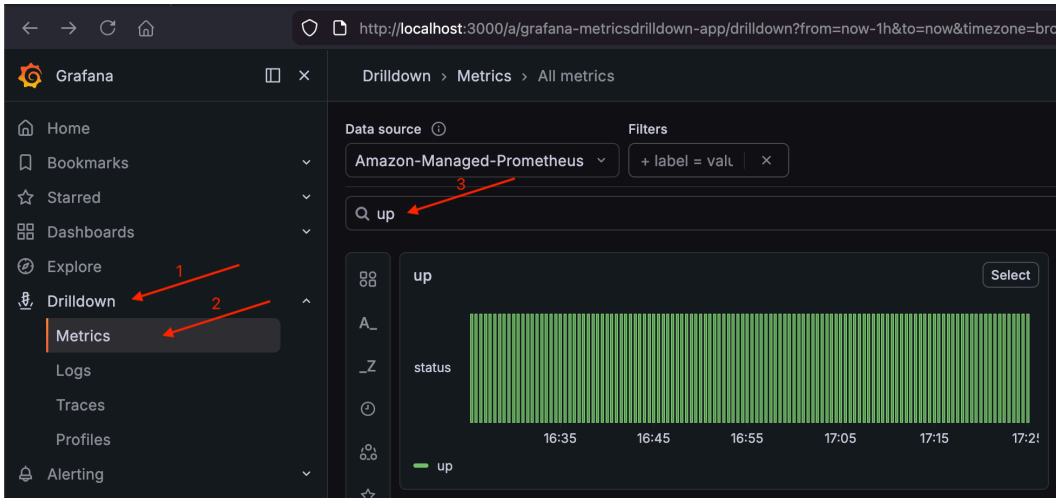
1. Vai a  Connessioni > Origini dati  e conferma che  Amazon-Managed-Prometheus  sia elencato come origine dati predefinita.

   Convalida l'origine dati AMP in Grafana




2. Vai a Drilldown > Metriche e cerca la metrica. up Dovresti vedere i risultati degli obiettivi di scrape del tuo cluster.

   Convalida la **up** metrica in Grafana




Se up vengono visualizzati i risultati, la pipeline (cluster → Prometheus → AMP → Grafana) funziona.

Convalida le metriche della GPU DCGM

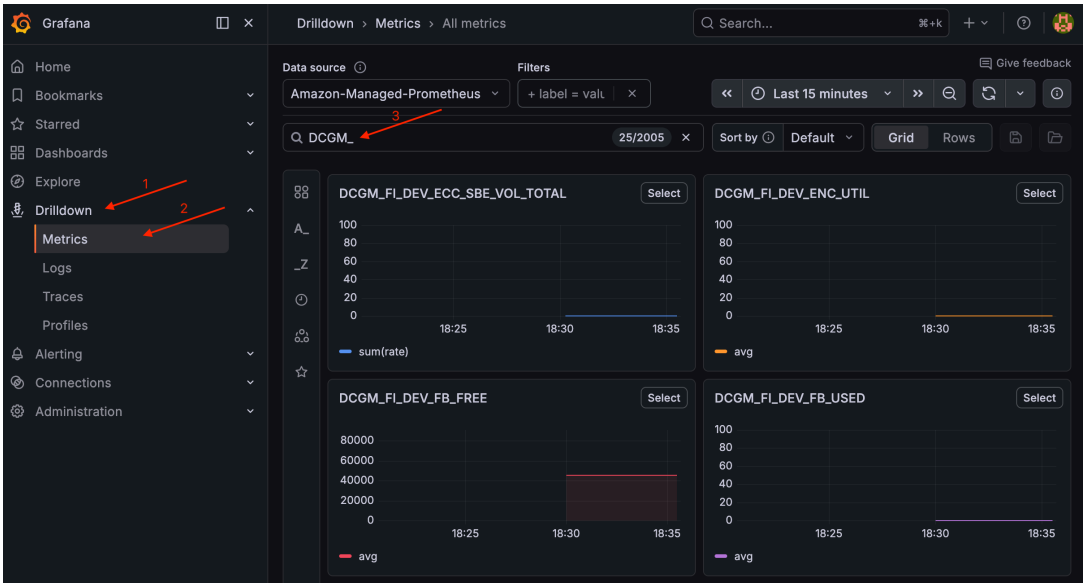
Il DCGM Exporter DaemonSet funziona su nodi GPU e riporta le metriche relative all'utilizzo della GPU, alla memoria, alla temperatura, al consumo energetico, alla larghezza di banda NVLink e all'attività tensoriale.

Verifica l'esportatore DCGM: DaemonSet

```
kubectl get daemonset dcgm-exporter -n monitoring
```

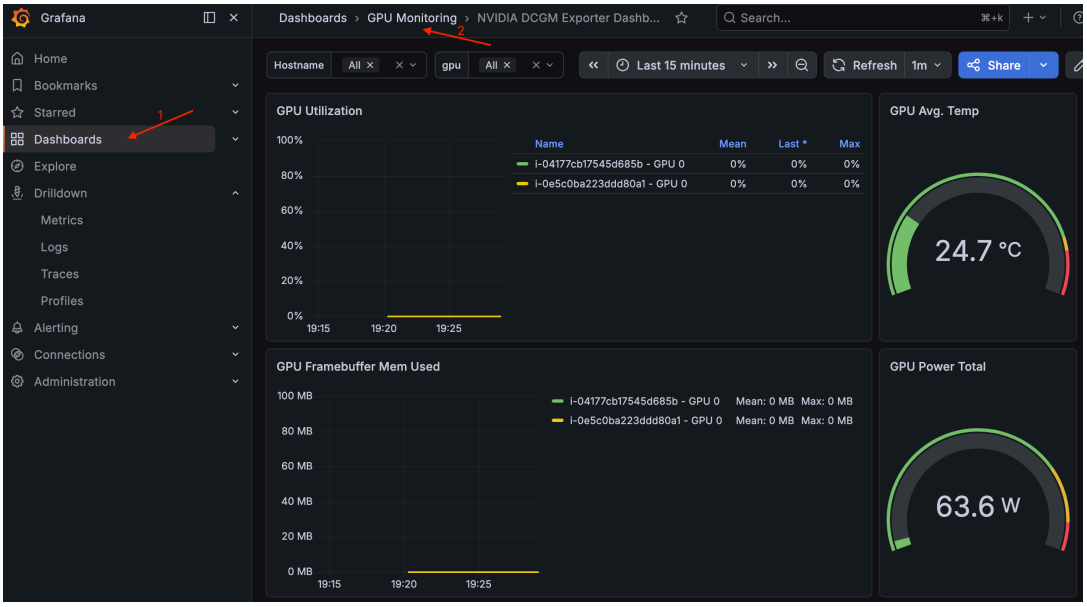
Una volta che un nodo GPU è in esecuzione (dal passaggio 2 o dal passaggio 4), dovresti vedere uno o più pod pronti. Per convalidare le metriche DCGM, accedi a Drilldown > Metrics in Grafana e cerca. `DCGM_`

Convalida le metriche DCGM in Grafana




Per visualizzare la dashboard, vai su Dashboard > Monitoraggio GPU > NVIDIA DCGM Exporter Dashboard.

Dashboard di NVIDIA DCGM Exporter a Grafana



## Il modello pesa un secchio S3

Terraform ha già creato un bucket Amazon S3 per memorizzare i pesi dei modelli, uno `model-storage-sa` ServiceAccount nel `default` namespace, una policy IAM con ambito per il bucket e un'EKS Pod Identity Association che li collega. I Workload Pod impostati possono leggere e scrivere nel bucket. `serviceAccountName: model-storage-sa`

Verifica il bucket

Recupera il nome del bucket dagli output di Terraform:

```
MODEL_BUCKET=$(terraform output -raw model_bucket)
echo ${MODEL_BUCKET}
```

Verifica che il bucket esista:

```
aws s3api head-bucket --bucket ${MODEL_BUCKET}
```

Output previsto:

```
{
    "BucketArn": "arn:aws:s3:::ai-eks-docs-models-20250612abc1",
    "BucketRegion": "us-east-2",
    "AccessPointAlias": false
}
```

Convalida l'accesso a S3 da un Pod

Esegui un Pod singolo con l'immagine AWS CLI, utilizzando `model-storage-sa` ServiceAccount, per confermare che EKS Pod Identity sia cablato e che l'accesso S3 funzioni:

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: s3-test
  labels:
    guide: ai-eks-docs
spec:
  serviceAccountName: model-storage-sa
  containers:
    - name: aws-cli
```

```
      image: public.ecr.aws/aws-cli/aws-cli:2.27.0
      command:
        - sh
        - -c
        - |
          echo "=== Caller Identity ==="
          aws sts get-caller-identity
          echo ""
          echo "=== S3 Write Test ==="
          echo "pod identity works" | aws s3 cp - s3://${MODEL_BUCKET}/test.txt
          echo ""
          echo "=== S3 List Test ==="
          aws s3 ls s3://${MODEL_BUCKET}/
          echo ""
          echo "=== S3 Delete Test ==="
          aws s3 rm s3://${MODEL_BUCKET}/test.txt
  restartPolicy: Never
EOF
```

Attendi il completamento del Pod e controlla i log:

```
kubectl wait --for=jsonpath='{.status.phase}'=Succeeded pod/s3-test --timeout=300s
kubectl logs s3-test
```

Output previsto:

```
=== Caller Identity ===
{
    "UserId": "AROA...:eks-ai-eks-docs-model-s-...",
    "Account": "123456789012",
    "Arn": "arn:aws:sts::123456789012:assumed-role/ai-eks-docs-models-.../eks-ai-eks-
docs-..."
}

=== S3 Write Test ===
upload: - to s3://ai-eks-docs-models-20250612abc1/test.txt

=== S3 List Test ===
2026-07-15 12:00:00         19 test.txt

=== S3 Delete Test ===
delete: s3://ai-eks-docs-models-20250612abc1/test.txt
```

L'identità del chiamante conferma che il Pod ha assunto il ruolo di archiviazione del modello tramite EKS Pod Identity. I comandi S3 confermano l'accesso in lettura e scrittura.

Pulisci il pod di prova:

```
kubectl delete pod s3-test
```

## Fasi successive

Con il cluster pronto, puoi passare a Load & Serve Model per implementare un modello linguistico di grandi dimensioni e interagire con l'endpoint di inferenza.

## Pulizia

> **Tip**
>
> Se intendi continuare con le sezioni successive di questa guida, salta la pulizia completa. Eseguila solo quando hai finito.

Elimina i carichi di lavoro di test in modo che nessun Pod contenga nodi GPU:

```
kubectl delete pod nvidia-smi --ignore-not-found
kubectl delete deployment gpu-overflow-test --ignore-not-found
```

Annulla la prenotazione della capacità senza distruggere il cluster

Se desideri solo rilasciare l'ODCR e tornare a Spot e On-Demand capacity, riporta la `nodepools` variabile alla `spot-ondemand` strategia:

```
terraform apply -var 'nodepools={"spot-ondemand"={}}'
```

Questo riduce `reserved` i requisiti relativi al NodePool tipo di capacità, distrugge l'ODCR e lascia il cluster, lo stack di monitoraggio e il bucket S3 al loro posto.

> **Important**
>
> L'annullamento di una prenotazione non comporta la chiusura delle istanze già in esecuzione su di essa. Tali istanze continuano a funzionare alle On-Demand tariffe standard fino alla loro

chiusura. Elimina prima i carichi di lavoro della GPU, come mostrato sopra, in modo che il nodo riservato si esaurisca prima del rilascio della prenotazione.

Distruggi il cluster e tutte le risorse Terraform-managed

Svuota i Karpenter-managed nodi prima di distruggerli, in modo che nessun ciclo di vita in corso blocchi la distruzione. Elimina quelli PodDisruptionBudgets che potrebbero impedire un drenaggio, quindi elimina: NodeClaims

```
kubectl delete pdb -A --all --ignore-not-found
kubectl delete nodeclaim --all --wait=true --timeout=900s
```

Quindi distruggi tutto ciò che Terraform ha creato, incluso il cluster EKS, il VPC, lo stack di monitoraggio, l' NodePools and NodeClasses, il bucket del modello S3 e qualsiasi ODCR:

```
terraform destroy
```

**Warning**

Il bucket S3 è stato creato con i pesi del modello`force_destroy = true`, quindi `terraform destroy` elimina il bucket insieme a tutti i pesi del modello che hai caricato su di esso. Copia prima tutto ciò che desideri conservare in un'altra posizione.

**Note**

Il repository fornisce anche un `scripts/cleanup.sh` helper che esegue i passaggi di drenaggio e distruzione precedenti e quindi analizza tutti i volumi EBS orfani contrassegnati con il nome del cluster. Eseguilo dall'interno della `terraform/<mode>/` directory da cui hai fatto la domanda e passa a saltare la richiesta di conferma `--auto-approve` di Terraform.

Verifica che la prenotazione sia finita

Conferma che non rimanga alcuna prenotazione di capacità attiva per il cluster:

```
aws ec2 describe-capacity-reservations \
```

```
  --filters "Name=state,Values=active" "Name=tag:nodepool,Values=reserved-spot-
ondemand" \
  --query 'CapacityReservations[].CapacityReservationId' \
  --output text
```

Un risultato vuoto significa che nessuna prenotazione è attiva e non si applicano ulteriori costi.

# Esegui carichi di lavoro di AI/ML inferenza su Amazon EKS

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS. AI/ML

L'inferenza è il processo di esecuzione di un modello di intelligenza artificiale addestrato per generare previsioni o risultati a partire dai dati di input. Ciò include la fornitura di modelli linguistici di grandi dimensioni (LLM) che producono testo o codice, modelli di diffusione che generano immagini, modelli vocali per la sintesi e la trascrizione vocale e modelli video per l'analisi o la generazione in tempo reale. Amazon EKS fornisce una soluzione scalabile e portatile per distribuire questi carichi di lavoro di inferenza con GPU NVIDIA o acceleratori AWS Trainium, scalabilità dinamica e integrazione con il più ampio panorama di strumenti e servizi di Kubernetes. AWS

## Perché usare Amazon EKS per l'inferenza

Amazon EKS combina l'orchestrazione di Kubernetes con l' AWS infrastruttura per affrontare le principali sfide dell'esecuzione dell'inferenza su larga scala:

- Scalabilità dinamica della GPU: Karpenter fornisce istanze GPU di dimensioni adeguate su richiesta in base alle richieste di risorse Pod, scalando da zero quando sono inattive e aggiungendo capacità all'aumentare del traffico. In questo modo si evita il provisioning eccessivo di costose risorse GPU.
- Avvio rapido a freddo: SOCI (Seekable OCI) scarica e decomprime contemporaneamente grandi livelli di immagini dei contenitori, riducendo i tempi di estrazione delle immagini da minuti a secondi. In combinazione con lo streaming dei modelli da Amazon S3 direttamente alla memoria GPU, i Pods possono iniziare a funzionare in meno di due minuti.
- Ripristino automatico dei guasti della GPU: l'agente di monitoraggio dei nodi EKS rileva i guasti hardware della GPU e attiva la sostituzione automatica dei nodi, riducendo al minimo i tempi di inattività senza intervento manuale.

- Ottimizzazione dei costi: le istanze Spot con On-Demand fallback, selezione delle istanze della dimensione corretta e comportamento scalabile a zero riducono i costi della GPU. La capacità riservata (ODCR) offre ulteriori risparmi per i carichi di lavoro stazionari.
- Standard aperti e portabilità: i carichi di lavoro di inferenza vengono eseguiti su API Kubernetes standard (Deployments, Services, HPA) con server modello open source come VLLM o SGLang, che garantiscono la portabilità tra gli ambienti.
- Monitoraggio integrato: le metriche Prometheus di VLLM o SGLang e NVIDIA DCGM Exporter forniscono visibilità sulla latenza delle richieste, sul throughput dei token, sull'utilizzo della GPU e sull'utilizzo della memoria tramite le dashboard Grafana.

## Glossario

In questa sezione vengono utilizzati i seguenti termini:

- Inferenza: processo di esecuzione di un modello addestrato per generare output (testo, incorporamenti, classificazioni) dai dati di input.
- Model server: un servizio containerizzato che carica un modello in memoria, riceve richieste di inferenza e restituisce previsioni. Gli esempi includono vLLM, SGlang, Triton Inference Server e Text Generation Inference (TGI). https://github.com/huggingface/text-generation-inference
- Pesi del modello: i parametri appresi di un modello addestrato, archiviati come file (in genere in formato GGUF) che il server dei modelli SafeTensors carica nella memoria della GPU.
- Acceleratore: hardware specializzato come GPU o AWS Trainium/Inferentia chip NVIDIA che velocizza le operazioni sulla matrice necessarie per l'inferenza.
- Parallelismo tensoriale: suddivisione di un modello su più GPU sullo stesso nodo per servire modelli che superano la memoria di una singola GPU.
- Cache KV: un buffer di memoria che memorizza le coppie chiave-valore calcolate in precedenza durante la generazione del testo, evitando calcoli ridondanti per ogni nuovo token.

## Come funziona l'inferenza su Amazon EKS

Ad alto livello, l'implementazione di un carico di lavoro di inferenza su Amazon EKS prevede i seguenti passaggi:

| Fase | Description |
| --- | --- |
| Configura il cluster | Crea un cluster EKS con GPU-enabled nodi, monitoraggio e archiviazione. Vedi Configurazione del cluster nei documenti AI/ML su EKS. |
| Memorizza i pesi dei modelli | Scarica i pesi dei modelli da un registro dei modelli (come Hugging Face) e memorizzali in Amazon S3 per caricarli rapidamente durante gli eventi di avvio a freddo o di scale-up. |
| Implementa il server modello | Crea una distribuzione Kubernetes che esegua un server modello (come vLLM) su nodi GPU. Il server modello trasmette i pesi da S3 alla memoria della GPU ed espone un'API. OpenAI-compatible |
| Esponi l'endpoint di inferenza | Crea un servizio Kubernetes per fornire un endpoint di rete stabile. Usa il port-forwarding per i test o un load balancer come AWS Application Load Balancer (ALB) per il traffico di produzione. |
| Monitoraggio e scalabilità | Usa le metriche di Prometheus (frequenza di richiesta, velocità effettiva dei token, latenza, utilizzo della cache KV) per monitorare le prestazioni e configurare la scalabilità automatica. |

## Metriche delle prestazioni di inferenza

La comprensione delle prestazioni di inferenza richiede il monitoraggio delle metriche relative a latenza e throughput:

Metriche di latenza

- Time to First Token (TTFT): tempo che intercorre tra l'arrivo della richiesta e il primo token generato. Fondamentale per le applicazioni interattive.
- Time Per Output Token (TPOT): tempo medio per generare ogni token successivo dopo il primo.
- End-to-end latenza della richiesta: tempo totale dall'invio della richiesta al completamento della risposta completa.

Metriche di produttività

- Richieste al secondo: richieste di inferenza totali fornite al secondo in tutte le repliche.

- Token di output al secondo: velocità di generazione dei token, misurata come token di output totali divisi per il tempo trascorso.
- Utilizzo della GPU: percentuale di cicli di calcolo della GPU utilizzati attivamente per l'inferenza.
- Utilizzo della cache KV: percentuale di memoria cache KV allocata in uso, che indica il livello massimo di capacità del server.

## Riduzione del tempo di avvio a freddo

I contenitori di inferenza di grandi dimensioni (8-15 GB) e il peso dei modelli (10-100+ GB) possono rallentare l'avvio del Pod. Le seguenti tecniche riducono al minimo i ritardi di avviamento a freddo:

- Pull parallelo SOCI: scarica e decomprime i livelli dell'immagine del contenitore contemporaneamente anziché in sequenza. Abilitato per impostazione predefinita in EKS Auto Mode per le istanze GPU.
- Streaming di modelli da S3: strumenti come Run:ai Model Streamer trasmettono i modelli direttamente da Amazon S3 alla memoria della GPU, bypassando il disco locale e riducendo il tempo di caricamento da minuti a secondi.
- Archivia le immagini in Amazon ECR: l'estrazione da un repository ECR regionale su un endpoint VPC evita la latenza Internet per le immagini di contenitori di grandi dimensioni.
- Memorizzazione nella cache dell'Instance Store: G-family le istanze con dischi NVMe locali possono memorizzare nella cache i layer di container e gli artefatti del modello, velocizzando i pull successivi sullo stesso nodo.

## Cosa implementerai

La procedura dettagliata del modello Load & Serve ti guida nella distribuzione di un'applicazione di inferenza end-to-end:

1. Scarica i pesi dei modelli: un Kubernetes Job scarica il Ministral-3-8B-Instruct modello da Hugging Face e lo carica nel bucket S3.
2. Implementa vLLM: una distribuzione esegue vLLM con Run:ai Model Streamer per trasmettere i pesi da S3 direttamente nella memoria della GPU, servendo un'API. OpenAI-compatible
3. Monitora con Grafana: A collega le metriche VLLM Prometheus allo stack di monitoraggio per creare dashboard in ServiceMonitor tempo reale.

4. Implementa un frontend di chat: Open WebUI fornisce un'interfaccia di chat basata su browser connessa all'endpoint vLLM.

La procedura dettagliata utilizza l'infrastruttura cluster della the section called “Configura il cluster” sezione e funziona sia con EKS Auto Mode che con i percorsi Karpenter autogestiti.

## Modelli Load & Serve su Amazon EKS

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS. AI/ML

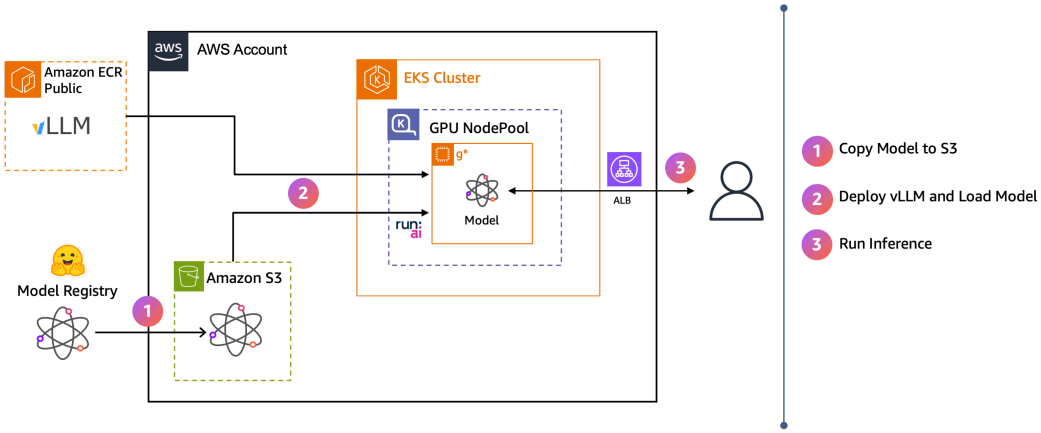
I passaggi descritti in questa sezione consentono di implementare un modello di linguaggio ampio (LLM) su Amazon EKS, utilizzarlo con vLLM e interagire con l'endpoint di inferenza.

La procedura dettagliata utilizza i seguenti strumenti:

- vLLM: un motore di inferenza ad alto rendimento ottimizzato per il servizio LLM e la gestione della memoria GPU.
- Run:ai Model Streamer: trasmette i pesi dei modelli direttamente da Amazon S3 alla memoria della GPU, riducendo il tempo di caricamento da minuti a secondi.
- Open WebUI: un frontend di chat ospitato autonomamente che si connette all'API di vLLM. OpenAI-compatible

Questa sezione utilizza il Ministral-3-8B-Instruct-2512 modello, ma puoi implementare qualsiasi modello di intelligenza artificiale supportato da vLLM. Per un elenco dei modelli supportati, vedi Modelli supportati nella documentazione di vLLM.

> **Important**
>
> Usa il cluster che hai creato nella the section called “Configura il cluster” sezione. Le istruzioni di questa procedura dettagliata sono valide sia per EKS Auto Mode che per Karpenter autogestito.




Il diagramma dell'architettura mostra il flusso end-to-end:

1. I pesi dei modelli vengono scaricati da Hugging Face su Amazon S3.
2. vLLM trasmette il modello direttamente da S3 alla memoria della GPU utilizzando Model Streamer. Run:ai
3. Gli utenti inviano richieste di inferenza all'endpoint vLLM.

Una volta completati questi passaggi, si dispone di un endpoint di inferenza vLLM che è possibile utilizzare per interagire con un modello Ministral tramite un'applicazione frontend di chat. Per ulteriori informazioni sull'ottimizzazione del tempo di caricamento del modello su Amazon EKS, consulta. the section called "Accelera il caricamento dei modelli"

## Prerequisiti

Completa i passaggi nella sezione Configurazione del cluster.

Se hai aperto un nuovo terminale, imposta il nome e la regione del cluster utilizzati nella sezione Configurazione del cluster tramite CLI:

```
export CLUSTER_NAME=ai-eks-docs
export AWS_REGION=us-east-2
```

Cerca il bucket dei pesi del modello che hai creato nella fase del bucket Model weights S3:

```
MODEL_BUCKET=$(aws s3api list-buckets \
  --query "Buckets[?starts_with(Name, '${CLUSTER_NAME}-models-')].Name | [0]" \
  --output text)
echo "Model bucket: ${MODEL_BUCKET}"
```

## Passaggio 1: scarica il modello da Hugging Face

In questo passaggio, distribuisci un Kubernetes Job che scarica il modello da Hugging Face e lo carica nel bucket S3 che hai creato nella sezione dei prerequisiti.

Per scaricare il modello, applica il seguente Job manifest:

Example Scarica il modello Job manifest

```
cat << EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: model-download
  namespace: default
  labels:
    guide: ai-eks-docs
spec:
  backoffLimit: 10
  activeDeadlineSeconds: 3600
  ttlSecondsAfterFinished: 86400
  template:
    spec:
      restartPolicy: Never
      serviceAccountName: model-storage-sa
      containers:
      - name: downloader
        image: python:3.11-slim
        command: ["/bin/bash", "-c"]
        args:
        - |
          set -e
          pip install -q huggingface_hub boto3
          echo "Downloading Ministral-3-8B-Instruct-2512 from Hugging Face..."
          python3 -c "from huggingface_hub import snapshot_download;
 snapshot_download('mistralai/Ministral-3-8B-Instruct-2512', local_dir='/tmp/
mistral', allow_patterns=['*.json', '*.txt', '*.md', 'consolidated.safetensors'],
 ignore_patterns=['model-*.safetensors', 'model.safetensors.index.json'])"
          echo "Uploading to S3 bucket: \${MODEL_BUCKET}"
          python3 << 'PYTHON'
          import boto3
          import os
          from pathlib import Path
```

```
          s3 = boto3.client('s3')
          bucket = os.environ.get('MODEL_BUCKET')
          local_dir = Path("/tmp/mistral")

          for file_path in local_dir.rglob("*"):
              if file_path.is_file():
                  if '.cache' in file_path.parts:
                      continue
                  s3_key = f"Ministral-3-8B-Instruct-2512/
{file_path.relative_to(local_dir)}"
                  print(f"Uploading {file_path.name}...")
                  s3.upload_file(str(file_path), bucket, s3_key)
          print("Upload complete!")
          PYTHON
        env:
        - name: MODEL_BUCKET
          value: "${MODEL_BUCKET}"
        - name: HF_HUB_DISABLE_XET
          value: "1"
        resources:
          requests:
            memory: "2Gi"
            cpu: "1"
          limits:
            memory: "4Gi"
            cpu: "2"
EOF
```

Attendi il completamento del lavoro. I pesi del modello (consolidated.safetensors) sono di circa 10,4 GB e questo passaggio richiede in genere 3-5 minuti.

```
kubectl wait --for=condition=complete job/model-download --timeout=600s
```

Output previsto:

```
job.batch/model-download condition met
```

Verifica che i pesi del modello siano stati caricati su S3:

```
aws s3 ls s3://$(kubectl get job model-download -o
 jsonpath='{.spec.template.spec.containers[0].env[?(@.name=="MODEL_BUCKET")].value}')/
Ministral-3-8B-Instruct-2512/ --recursive
```

Output previsto:

```
2026-05-18 10:29:53      20311 Ministral-3-8B-Instruct-2512/README.md
2026-05-18 10:29:53       2361 Ministral-3-8B-Instruct-2512/SYSTEM_PROMPT.txt
2026-05-18 10:29:53       1903 Ministral-3-8B-Instruct-2512/config.json
2026-05-18 10:29:54 10420633176 Ministral-3-8B-Instruct-2512/consolidated.safetensors
2026-05-18 10:29:53        131 Ministral-3-8B-Instruct-2512/generation_config.json
2026-05-18 10:29:53       1185 Ministral-3-8B-Instruct-2512/params.json
2026-05-18 10:29:53        976 Ministral-3-8B-Instruct-2512/processor_config.json
2026-05-18 10:29:53   16753777 Ministral-3-8B-Instruct-2512/tekken.json
2026-05-18 10:29:53   17077402 Ministral-3-8B-Instruct-2512/tokenizer.json
2026-05-18 10:29:53      21168 Ministral-3-8B-Instruct-2512/tokenizer_config.json
```

Il file consolidated.safetensors contiene i pesi del modello (circa 10,4 GB). I file rimanenti sono file di configurazione e tokenizer necessari a vLLM per servire il modello.

## Fase 2: Implementare il contenitore di inferenza

In questa sezione, distribuisci vLLM come distribuzione Kubernetes per servire il modello che hai caricato su Amazon S3.

Questa sezione utilizza AWS Deep Learning Containers (DLC), che sono immagini Docker preinstallate con framework di deep learning e ottimizzate per le prestazioni sull'infrastruttura. AWS I DLC includono patch di sicurezza, versioni del framework convalidate e configurazioni ottimizzate dei driver GPU.

Questa distribuzione utilizza il seguente AWS DLC per vLLM 0.21.0 con supporto SOCI:. https://gallery.ecr.aws/deep-learning-containers/vllm `public.ecr.aws/deep-learning-containers/vllm:0.21.0-gpu-py312-cu130-ubuntu22.04-ec2-v1.0-soci`

Il tag image indica vLLM 0.21.0 con supporto GPU, Python 3.12, CUDA 13.0, Ubuntu 22.04, ottimizzato per i carichi di lavoro e per un avvio più rapido del container. EC2-based SOCI-enabled

Questo manifest crea una distribuzione che esegue vLLM su un nodo GPU e trasmette il modello direttamente da S3 nella memoria della GPU utilizzando Model Streamer. Run:ai Il manifest crea anche un servizio ClusterIP che espone l'endpoint vLLM sulla porta 8000 per l'accesso all'interno del cluster.

Per ulteriori informazioni sull'ottimizzazione del tempo di caricamento del modello su Amazon EKS, consulta. the section called "Accelera il caricamento dei modelli" L'esempio seguente viene utilizzato `--enforce-eager` per accelerare i tempi di caricamento in uno scenario introduttivo. Si consiglia

di utilizzare altre tecniche per accelerare i tempi di caricamento del modello, come descritto inthe section called "Il compromesso --enforce-eager".

Applica il manifesto:

Example Distribuzione e servizio vLLM YAML

```
cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vllm-inference-app
  labels:
    guide: ai-eks-docs
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vllm-inference-app
  template:
    metadata:
      labels:
        app: vllm-inference-app
        guide: ai-eks-docs
    spec:
      serviceAccountName: model-storage-sa
      tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
      nodeSelector:
        karpenter.sh/nodepool: gpu-inf
      containers:
      - name: vllm-inference
        image: public.ecr.aws/deep-learning-containers/vllm:0.21.0-gpu-py312-cu130-
ubuntu22.04-ec2-v1.0-soci
        ports:
        - containerPort: 8000
        args:
        - "--model=s3://${MODEL_BUCKET}/Ministral-3-8B-Instruct-2512/"
        - "--host=0.0.0.0"
        - "--port=8000"
        - "--tensor-parallel-size=1"
        - "--gpu-memory-utilization=0.9"
```

```
        - "--max-model-len=8192"
        - "--max-num-seqs=128"
        - "--load-format=runai_streamer"
        - "--enforce-eager"
        - "--tokenizer_mode=mistral"
        - "--config_format=mistral"
        - "--enable-auto-tool-choice"
        - "--tool-call-parser=mistral"
        resources:
          limits:
            nvidia.com/gpu: 1
          requests:
            memory: "40Gi"
            cpu: "8"
---
apiVersion: v1
kind: Service
metadata:
  name: vllm-inference-svc
  namespace: default
  labels:
    app: vllm-inference-app
spec:
  selector:
    app: vllm-inference-app
  ports:
  - name: http
    port: 8000
    targetPort: 8000
    protocol: TCP
EOF
```

Verifica che il pod vLLM sia nello stato Pronto:

```
kubectl get pod -l app=vllm-inference-app -w
```

Output previsto:

```
NAME                                  READY   STATUS    RESTARTS   AGE
vllm-inference-app-65df5fddc8-5kmjm   1/1     Running   0          86s
```

Potrebbero essere necessari circa 2 minuti prima che l'immagine del contenitore venga estratta e che vLLM trasmetta i pesi dei modelli da S3 alla memoria della GPU. Attendi che il pod venga visualizzato 1/1 nella colonna READY prima di procedere.

La combinazione di EKS, SOCI e Run:ai Model Streamer consente l'avvio rapido del pod. Per controllare l'ora di avvio di ogni fase, visualizza gli eventi del pod:

```
kubectl describe pod -l app=vllm-inference-app | grep -A 20 "Events:"
```

Output previsto:

```
Events:
  Type     Reason            Age   From                   Message
  ----     ------            ----  ----                   -------
  Warning  FailedScheduling  86s   default-scheduler      0/2 nodes are available: 2
 node(s) had untolerated taint(s).
  Normal   Nominated         85s   eks-auto-mode/compute  Pod should schedule on:
 nodeclaim/gpu-inf-kqkq6
  Normal   Scheduled         55s   default-scheduler      Successfully assigned
 default/vllm-inference-app-d9d54586d-csmd7 to i-04f8792414384d2d3
  Normal   Pulling           52s   kubelet                Pulling image
 "public.ecr.aws/deep-learning-containers/vllm:0.21.0-gpu-py312-cu130-ubuntu22.04-ec2-
v1.0-soci"
  Normal   Pulled            4s    kubelet                Successfully pulled image
 "public.ecr.aws/deep-learning-containers/vllm:0.21.0-gpu-py312-cu130-ubuntu22.04-ec2-
v1.0-soci" in 48.376s (48.376s including waiting). Image size: 8802823997 bytes.
  Normal   Created           4s    kubelet                Created container vllm-
inference
  Normal   Started           4s    kubelet                Started container vllm-
inference
```

In questo esempio, il provisioning del nodo GPU è stato eseguito in 30 secondi e l'immagine del contenitore da 8,8 GB è stata estratta in circa 48 secondi utilizzando SOCI. L'estrazione rapida delle immagini riduce i tempi di avvio a freddo per i contenitori di inferenza di grandi dimensioni, il che consente di scalare i pod GPU in modo dinamico invece di sovra-provisionare la capacità della GPU inattiva.

Quindi, controllate i log vLLM per verificare il tempo di caricamento del modello:

```
kubectl logs $(kubectl get pod -l app=vllm-inference-app -o
 jsonpath='{.items[0].metadata.name}') | grep -i 'Model loading took'
```

Output previsto:

```
INFO 05-18 18:41:49 [gpu_model_runner.py:4959] Model loading took 9.81 GiB memory and
 5.023344 seconds
```

Il log conferma che Run:ai Model Streamer ha caricato i pesi del modello da 10,4 GB direttamente da S3 nella memoria della GPU in circa 5 secondi, consumando 9,8 GiB di memoria GPU.

Il tempo di download dell'immagine in questo esempio è stato eseguito utilizzando un'istanza g6e.4xlarge, con una larghezza di banda di rete sostenuta di 20 Gbps. L'estrazione delle immagini e i tempi di caricamento del modello varieranno in altri tipi di istanza a seconda della larghezza di banda di rete disponibile.

## Fase 3: Eseguire l'inferenza

Con vLLM Deployment in esecuzione, convalida l'endpoint di inferenza e implementa un frontend di chat per interagire con il modello.

Esegui un test di convalida del modello

Esponi l'endpoint di inferenza tramite port forward:

```
kubectl port-forward svc/vllm-inference-svc 8000:8000
```

Apri una nuova finestra di terminale, quindi verifica che il contenitore di inferenza stia rispondendo:

```
curl -sI -X GET http://localhost:8000/health
```

Output previsto:

```
HTTP/1.1 200 OK
date: Fri, 18 May 2026 00:39:23 GMT
server: uvicorn
content-length: 0
```

## Fase 4: Monitorare vLLM

vLLM espone le metriche di Prometheus pronte all'uso, tra cui la frequenza di richiesta, il throughput dei token, la latenza end-to-end e l'utilizzo della cache GPU KV. In questa sezione, puoi utilizzare

queste metriche con lo stack di monitoraggio impostato nelle fasi di configurazione del cluster e visualizzarle su una dashboard Grafana preconfigurata. ???

> **Important**
>
> È necessario completare la sottosezione Monitoraggio della sezione Configurazione del cluster tramite CLI prima di continuare. Questo passaggio dipende dal kube-prometheus-stack da installare e dal dashboard vLLM Grafana già fornito nel file dei valori.

### Applica il vLLM ServiceMonitor

A ServiceMonitor indica a Prometheus dove raccogliere le metriche vLLM.

```
cat << EOF | kubectl apply -f -
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: vllm-inference-app
  namespace: default
  labels:
    release: kube-prometheus-stack
spec:
  selector:
    matchLabels:
      app: vllm-inference-app
  endpoints:
  - port: http
    path: /metrics
    interval: 15s
EOF
```

Verifica che ServiceMonitor sia stato creato:

```
kubectl get servicemonitor vllm-inference-app
```

Output previsto:

```
NAME                  AGE
vllm-inference-app    5s
```

### Genera traffico di inferenza

Per riempire la dashboard con le metriche, genera traffico di inferenza rispetto all'endpoint vLLM già esposto tramite port-forward nella fase di convalida.

Scopri il nome del modello servito:

```
MODEL_NAME=$(curl -s http://localhost:8000/v1/models | jq -r '.data[0].id')
echo "Using model: $MODEL_NAME"
```

Invia 50 richieste di completamento della chat in parallelo:

```
for i in $(seq 1 50); do
  curl -s -X POST http://localhost:8000/v1/chat/completions \
    -H "Content-Type: application/json" \
    -d "{\"model\": \"$MODEL_NAME\", \"messages\": [{\"role\": \"user\", \"content\":
 \"Write a short poem about Kubernetes.\"}], \"max_tokens\": 128}" \
    > /dev/null &
done
wait
```

Mentre il traffico scorre (o subito dopo), controlla le metriche relative al throughput dei token direttamente dall'endpoint vLLM: `/metrics`

```
curl -s http://localhost:8000/metrics | grep -E '^vllm:(prompt_tokens_total|
generation_tokens_total|avg_generation_throughput_toks_per_s|
avg_prompt_throughput_toks_per_s)' | head
```

Le `vllm:generation_tokens_total` metriche `vllm:prompt_tokens_total` e sono numeri che aumentano in modo monotono i token di input e output serviti. Le `vllm:avg_generation_throughput_toks_per_s` metriche `vllm:avg_prompt_throughput_toks_per_s` e sono indicatori di produttività medi. Queste stesse metriche sono alla base della dashboard di Grafana che apri nella sottosezione seguente.

### Visualizza la dashboard di vLLM Grafana

Il file dei valori kube-prometheus-stack della sezione Monitoring fornisce già il dashboard vLLM della community (GNetID 25263) nella cartella GPU Monitoring, quindi non è necessaria alcuna importazione aggiuntiva.

Accedi a Grafana tramite il load balancer che hai impostato nella sezione Access Grafana. ??? Stampa il suo URL:

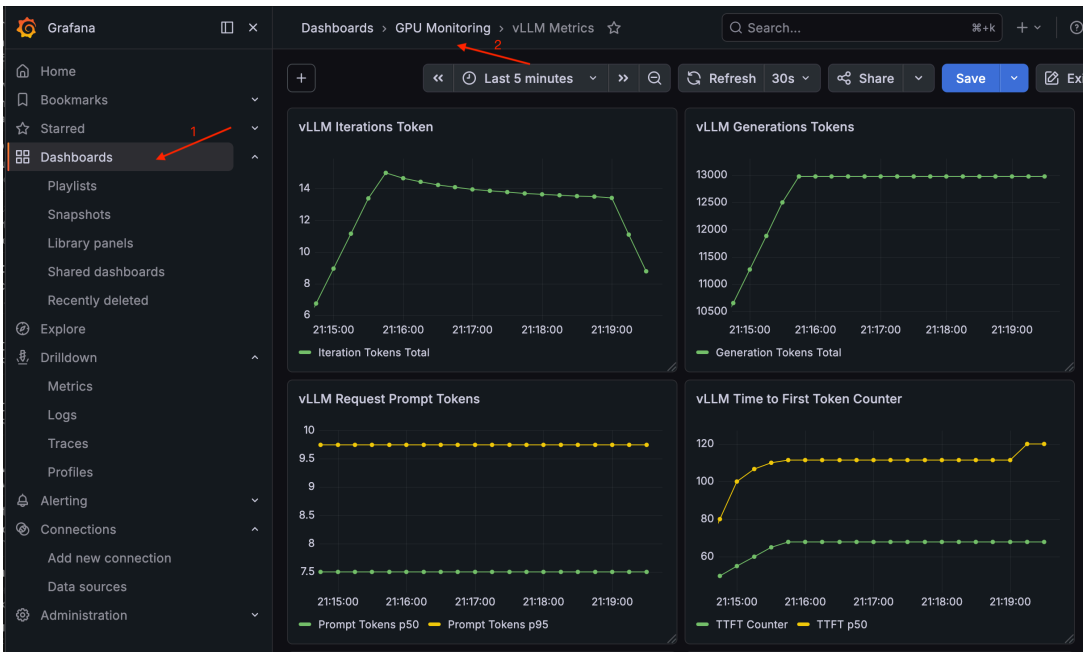
```
echo "http://$(kubectl get ingress kube-prometheus-stack-grafana -n monitoring -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')"
```

Apri l'URL nel tuo browser e accedi con nome utente `admin` e password con il seguente comando:

```
kubectl --namespace monitoring get secrets kube-prometheus-stack-grafana -o
 jsonpath="{.data.admin-password}" | base64 -d ; echo
```

Accedi a  Dashboard > Monitoraggio GPU > Metriche vLLM.

Pannello di controllo vLLM Grafana




La dashboard mostra la frequenza di richiesta, il throughput dei token di richiesta e generazione, i percentili di latenza e l'utilizzo della cache GPU KV per l'endpoint di inferenza vLLM.

## Fase 5: Implementa l'applicazione di chat

In questo passaggio, si implementa Open WebUI come frontend di chat per interagire con il modello. Open WebUI è un'interfaccia AI open source e ospitata autonomamente che supporta le OpenAI-compatible API e fornisce un'interfaccia di chat con cronologia delle conversazioni e rendering markdown. Poiché vLLM espone un' OpenAI-compatible API, Open WebUI si connette direttamente ad essa come backend.

Per distribuire l'applicazione Open WebUI, applica il seguente manifest:

Example Apri WebUI Deployment and Service YAML

```
cat << 'EOF' | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: open-webui
  namespace: default
  labels:
    app: open-webui
    guide: ai-eks-docs
spec:
  replicas: 1
  selector:
    matchLabels:
      app: open-webui
  template:
    metadata:
      labels:
        app: open-webui
        guide: ai-eks-docs
    spec:
      containers:
      - name: open-webui
        image: ghcr.io/open-webui/open-webui:v0.9.2
        ports:
        - containerPort: 8080
        resources:
          requests:
            cpu: "500m"
            memory: "500Mi"
          limits:
            cpu: "1000m"
            memory: "2Gi"
        env:
        - name: OPENAI_API_BASE_URLS
          value: "http://vllm-inference-svc:8000/v1"
        - name: OPENAI_API_KEY
          value: "dummy"
        - name: WEBUI_AUTH
          value: "False"
        - name: ENABLE_OLLAMA_API
```

```
          value: "False"
        - name: ENABLE_EVALUATION_ARENA_MODELS
          value: "False"
        - name: RAG_EMBEDDING_ENGINE
          value: ""
        volumeMounts:
        - name: webui-volume
          mountPath: /app/backend/data
      volumes:
      - name: webui-volume
        emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: open-webui
  namespace: default
  labels:
    app: open-webui
spec:
  type: ClusterIP
  selector:
    app: open-webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
EOF
```

Attendi che il pod Open WebUI sia pronto:

```
kubectl wait --for=condition=ready pod -l app=open-webui --timeout=300s
```

Output previsto:

```
pod/open-webui-6cbfc9867f-jf9w9 condition met
```

Per accedere all'applicazione, esponi Open WebUI tramite un AWS Application Load Balancer (ALB) connesso a Internet utilizzando quello creato in Configura il bilanciamento del carico. `alb IngressClass` ???

⚠ Open WebUI è accessibile pubblicamente senza autenticazione

Open WebUI funziona con l'autenticazione disattivata (`WEBUI_AUTH: "False"`), quindi chiunque raggiunga il load balancer ottiene un'interfaccia di chat non autenticata supportata dall'endpoint di inferenza GPU e può consumare la capacità della GPU. Gli scanner automatici trovano i sistemi di bilanciamento del carico pubblici in pochi minuti. È necessario limitare l'accesso con l'`alb.ingress.kubernetes.io/inbound-cidrs`annotazione e considerare l'elenco degli indirizzi IP di origine come una protezione minima piuttosto che completa. Per una postura più sicura, utilizza uno schema interno, aggiungi un certificato TLS e abilita l'autenticazione Open WebUI.

Trova il tuo indirizzo IP pubblico e memorizzalo come CIDR: `/32`

```
export MY_CIDR="$(curl -s https://checkip.amazonaws.com)/32"
echo $MY_CIDR
```

Il risultato sembra`203.0.113.4/32`. Se la rete assegna gli indirizzi in modo dinamico, l'indirizzo IP può cambiare, nel qual caso potrebbe essere necessario un intervallo più ampio, ad esempio. `203.0.113.0/24`

Applica quanto segue `Ingress` per creare l'ALB:

```
cat << EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: open-webui
  namespace: default
  labels:
    guide: ai-eks-docs
  annotations:
    alb.ingress.kubernetes.io/load-balancer-name: ai-eks-docs-chat-app
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/inbound-cidrs: ${MY_CIDR}
    alb.ingress.kubernetes.io/healthcheck-path: /health
spec:
  ingressClassName: alb
  rules:
```

```
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: open-webui
            port:
              number: 80
EOF
```

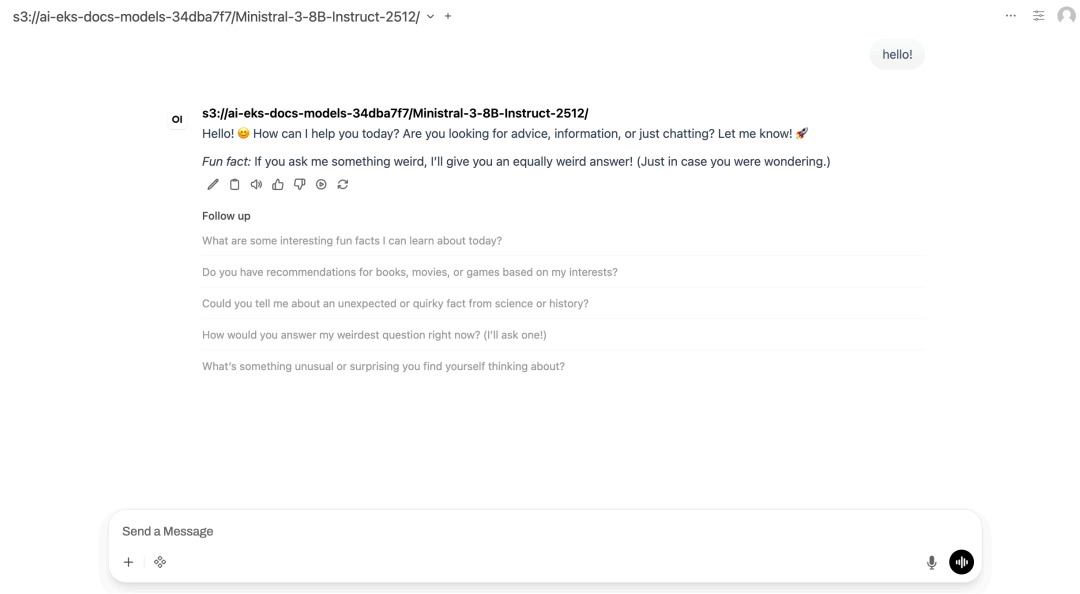
> **Percorso di Controllo dell'integrità**
>
> L'`healthcheck-path`annotazione indica i controlli di integrità del load balancer all'`/health`endpoint Open WebUI, poiché il matcher di controllo dello stato predefinito di ALB prevede una risposta di 200.

Il load balancer viene creato in modo asincrono e richiede uno o due minuti. Stampa l'URL:

```
echo "http://$(kubectl get ingress open-webui -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')"
```

Apri l'URL nel tuo browser. Viene visualizzata l'interfaccia di chat e puoi interagire con il modello Ministral.

> **Port-forwarding in alternativa**
>
> Port-forwarding (`kubectl port-forward svc/open-webui 8080:80`) rimane un'opzione per i test locali senza il provisioning di un load balancer.




## Eliminazione

Per rimuovere le risorse del carico di lavoro che hai creato in questa sezione, elimina l'applicazione Open WebUI, il server di inferenza vLLM e il Job per il download del modello:

```
kubectl delete ingress open-webui
kubectl delete deployment open-webui
kubectl delete service open-webui
kubectl delete deployment vllm-inference-app
kubectl delete service vllm-inference-svc
kubectl delete servicemonitor vllm-inference-app
kubectl delete job model-download
```

Per istruzioni sulla rimozione delle risorse dell'infrastruttura come il cluster e il bucket S3 NodePool, vedi Cluster Setup Cleanup. ???

# Inferenza del modello AI con scalabilità automatica sulle GPU con Amazon EKS

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Questa sezione descrive i concetti e le procedure per la scalabilità automatica dei carichi di lavoro di inferenza AI su GPU con Amazon EKS.

La singola replica del modello VLLm che hai distribuito nella precedente sezione Load & Serve Models può soddisfare un numero limitato di richieste simultanee prima che le nuove richieste inizino ad attendere, il che aumenta la latenza e può causare timeout delle richieste. Per stare al passo con il traffico, puoi scalare automaticamente l'implementazione orizzontalmente. La scalabilità automatica aggiunge repliche del carico di lavoro all'aumentare della domanda e le rimuove quando la domanda diminuisce, in modo da evitare l'eccessivo provisioning della capacità della GPU.

La scalabilità automatica delle inferenze avviene in due fasi.

- Scalabilità dei nodi: quando lo scheduler Kubernetes non ha spazio per posizionare nuovi pod sui nodi esistenti, i pod rimangono. `Pending` Karpenter risponde ai `Pending` pod fornendo nodi GPU adatti ad essi e rimuove i nodi quando le repliche del carico di lavoro diminuiscono.
- Scalabilità tramite pod: l'Horizontal Pod Autoscaler (HPA) monitora una metrica configurata. Quando tale metrica supera una soglia configurata, HPA aumenta il numero di repliche del carico di lavoro e vengono creati nuovi pod VLLm.

La fase di scalabilità dei nodi fornisce la capacità della GPU necessaria per l'esecuzione delle repliche del carico di lavoro. La fase di scalabilità basata su pod regola il numero di repliche del carico di lavoro per mantenere la metrica scelta vicina all'obiettivo.

## Provisioning dei nodi GPU con Karpenter

Karpenter gestisce il ciclo di vita dei nodi, effettuando il provisioning dei nodi quando i carichi di lavoro ne hanno bisogno e consolidando o rimuovendo i nodi quando non lo richiedono. Funziona con lo scheduler Kubernetes anziché sostituirlo. Lo scheduler posiziona i pod sui nodi. Karpenter fornisce la capacità per i pod che lo scheduler non può contenere, quindi consente allo scheduler di associarli una volta che il nodo è presente. `Ready`

Karpenter gestisce i nodi GPU in modo diverso rispetto a una flotta di nodi CPU.

Le GPU sono richieste come unità intere

A meno che non si utilizzi una tecnica di condivisione della GPU come time-slicing o MIG, un pod riceve una o più GPU intere anziché una frazione come farebbe con CPU o memoria. Ogni GPU ha la propria memoria dedicata, quindi le repliche sullo stesso nodo non condividono la memoria della GPU. Il modo in cui questo viene mappato ai nodi dipende dal tipo di istanza:

- Single-GPU instance (ad esempio`g6.xlarge`) — Il nodo contiene esattamente una replica. Le repliche, le GPU e i nodi sono scalabili uno a uno e ogni replica è isolata sulla propria istanza.
- Multi-GPU istanza (ad esempio`g6.12xlarge`, che ha 4 GPU): il plug-in del dispositivo pubblicizza il numero completo di GPU, quindi Karpenter può raggruppare diverse repliche su un nodo, una per GPU.
- Multi-GPU pod: un pod può richiedere più di una GPU. Ad esempio, imposta lo shard `nvidia.com/gpu: 2` di un modello di grandi dimensioni su due GPU e Karpenter posiziona il pod su un nodo con così tante GPU libere.

Karpenter ottimizza la disponibilità e i costi della GPU

Karpenter bilancia l'utilizzo della scarsa capacità della GPU con una gestione economicamente vantaggiosa:

- Selezione delle istanze: data la flessibilità NodePool, Karpenter seleziona il tipo di istanza più economico adatto ai pod. `Pending` Quando un avvio raggiunge una capacità insufficiente, riprova automaticamente altri tipi di istanza e zone di disponibilità, il che aumenta le probabilità di esaurimento della capacità della GPU.
- Consolidamento: quando la domanda diminuisce e le repliche aumentano, Karpenter recupera i nodi liberati. `WhenEmptyOrUnderutilized`rimuove i nodi vuoti e riconfeziona le repliche su un numero inferiore di nodi, mentre `WhenEmpty` recupera un nodo solo dopo che non è stato pianificato nulla su di esso.
- Protezione dei pod in volo: per evitare che i pod vengano interrotti durante il consolidamento, aggiungi l'`karpenter.sh/do-not-disrupt`annotazione, imposta un PodDisruptionBudget o utilizza l'impostazione per ritardare il consolidamento. `consolidateAfter`

Una volta che Karpenter ha raggiunto la capacità della GPU, prende il sopravvento la fase di scalabilità dei pod: HPA aggiunge e rimuove le repliche in base alla metrica configurata. Il resto di questa sezione spiega come scegliere quella metrica di scalabilità per i carichi di lavoro di inferenza.

## Scalabilità automatica tra GPU e CPU

In Kubernetes, i CPU-based carichi di lavoro tradizionali vengono scalati automaticamente immediatamente, ma l'inferenza no. GPU-based Esistono tre differenze principali:

Disponibilità delle metriche per la scalabilità automatica

Kubernetes fornisce metriche di CPU e memoria a HPA immediatamente

Il kubelet su ciascun nodo riporta l'utilizzo della CPU e della memoria all'API Kubernetes Metrics, in modo che HPA possa leggere questi valori direttamente e scalare i carichi di lavoro senza componenti aggiuntivi.

I carichi di lavoro accelerati richiedono metriche, configurazioni, esportatori e adattatori aggiuntivi

Il kubelet non riporta le metriche di GPU o di inferenza. Per scalare GPU-based l'inferenza, è necessario aggiungere gli esportatori che pubblicano i segnali pertinenti, come le metriche delle applicazioni VLLm e le metriche GPU dell'esportatore DCGM, e gli adattatori che espongono tali metriche a HPA.

Allocazione e condivisione delle risorse

I carichi di lavoro tradizionali condividono CPU e memoria in frazioni

I cgroup Linux dividono CPU e memoria in risorse frazionarie, permettendo a più pod di condividere lo stesso nodo. Un pod può richiedere una parte della capacità di un nodo, ad esempio 100 millicore di CPU o 512 MiB di memoria.

I carichi di lavoro accelerati allocano e condividono le GPU tramite un framework di allocazione GPU

Le GPU non sono condivise tramite cgroup come CPU e memoria. La configurazione Kubernetes più comune oggi utilizza il plug-in per dispositivi NVIDIA, che alloca intere GPU ai pod (ad esempio,). `nvidia.com/gpu: 1` Dynamic Resource Allocation (DRA) è una nuova API Kubernetes che alloca le GPU tramite dichiarazioni di risorse, offrendo una pianificazione e una gestione delle risorse più flessibili. Sia il plug-in del dispositivo che il DRA possono supportare tecniche di condivisione della GPU. Time-slicing consente a più pod di condividere una GPU interlacciando l'accesso nel tempo, mentre GPU (MIG) partiziona le Multi-Instance GPU supportate, come NVIDIA A100 e H100, in istanze GPU isolate dall'hardware.

Segnali di scalabilità

I carichi di lavoro tradizionali in genere si adattano all'utilizzo della CPU e della memoria

L'utilizzo è il segnale di scalabilità automatica standard per i carichi di lavoro di CPU e memoria. È un indicatore di ritardo perché aumenta solo dopo che il carico di lavoro è già sotto carico, ma per molte applicazioni tradizionali è sufficiente per prendere decisioni sulla scalabilità.

I carichi di lavoro accelerati dovrebbero essere scalabili in base ai principali indicatori, non all'utilizzo della GPU

> L'utilizzo della GPU (`DCGM_FI_DEV_GPU_UTIL`) indica se la GPU è occupata, non quanto è occupata. Poiché un server di inferenza come vLLM può rimanere vicino al 100% su un'ampia gamma di richieste in entrata, l'utilizzo è un segnale di scalabilità automatica inaffidabile. Scala invece in base agli indicatori principali che aumentano man mano che la domanda si avvicina alla capacità, come la profondità della coda delle richieste, il time to first token (TTFT) e la latenza delle richieste.

Di queste tre differenze sopra descritte, valuta attentamente i tuoi segnali di scalabilità. Poiché l'utilizzo della GPU non è un segnale ad alta confidenza della capacità di inferenza residua, è necessario identificare e raccogliere segnali che tengano traccia della domanda effettiva sul server di inferenza del modello.

La sezione successiva illustra quali segnali utilizzare come metriche di scalabilità e come interagiscono tra loro.

## Scelta delle metriche di scalabilità

Nessuna metrica indica quando scalare le repliche di inferenza del modello, quindi la scalabilità si basa su una combinazione di segnali provenienti dal server di inferenza. Questi segnali interagiscono, ognuno dei quali si basa sul precedente per catturare la domanda prima e in modo più affidabile rispetto a qualsiasi metrica singola:

| Signal | Metrica | Ruolo nella scalabilità |
|---|---|---|
| Queue | Profondità della coda delle richieste (numero di richieste in attesa di essere elaborate) | Un indicatore principale e il miglior trigger predefinito, perché nel momento in cui la domanda supera la capacità si forma una coda. |
| Latenza | Latenza di richiesta (latenza p95 o TTFT) | Attiva la scalabilità quando le risposte sono lente anche se la coda non si è ancora creata, mantenendo la latenza rivolta all'utente entro l'obiettivo. |

| Signal | Metrica | Ruolo nella scalabilità |
| --- | --- | --- |
| Memoria GPU | Utilizzo della cache KV (quanto è piena la memoria di richiesta della GPU) | Protegge dall'esaurimento della memoria della GPU; quando si avvicina al limite, il server inizia a rifiutare o a mettere in coda le richieste. |

Insieme costituiscono diversi livelli di difesa: la profondità della coda delle richieste è il fattore scatenante principale che si attiva nel momento in cui la domanda supera la capacità, la latenza delle richieste consente di controllare la lentezza delle risposte prima che si formi una coda e l'utilizzo della cache KV funge da backstop che impedisce l'esaurimento della memoria della GPU.

### Mettere tutto in pratica

Le seguenti sottosezioni mostrano come far funzionare questi segnali con uno specifico autoscaler.

- Trova le soglie metriche di scalabilità. Esegui il test di carico di una singola replica VLLm per individuarne il limite di capacità e le soglie di profondità e latenza della coda su cui basare la scalabilità. Inizia da qui.
- Scalabilità con HPA e KEDA. Repliche VLLm in scala automatica con KEDA e Horizontal Pod Autoscaler, utilizzando la profondità della coda delle richieste e la latenza end-to-end come segnali di scalabilità.

## Identifica le soglie metriche di scalabilità per l'inferenza AI

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS. AI/ML

Questa sezione mostra come eseguire il test di carico di un server di inferenza vLLM su Amazon EKS per identificare quando diventa saturo. Usa i risultati del test di carico per configurare la profondità della coda e le soglie di latenza per la scalabilità automatica orizzontale nella sezione successiva. the section called "HPA e KEDA"

La procedura dettagliata utilizza i seguenti strumenti:

- k6 (Grafana k6) è uno strumento di test di carico open source, eseguito come immagine del `grafana/k6` contenitore, che invia richieste di inferenza alla replica a frequenze di richiesta controllate.
- Ministral-3-8B-Instruct-2512 modello su un'istanza. `g6e.4xlarge` È possibile applicare la stessa metodologia a qualsiasi modello open source e tipo di GPU.

Prerequisiti

Questa sezione si basa su due sezioni precedenti della documentazione. Completali entrambi prima di iniziare:

- Cluster Setup fornisce un cluster EKS con una GPU NodePool, lo stack di monitoraggio kube-prometheus-stack (Prometheus e Grafana) e un bucket Amazon S3 per i pesi dei modelli. ???
- Load & Serve Models implementa `vllm-inference-app` ServiceMonitor il server di inferenza vLLM e crea il.

Conferma ogni elemento di seguito prima di iniziare e completa prima la sezione collegata se un controllo fallisce.

Se hai aperto un nuovo terminale, imposta il nome del cluster e la regione che hai usato in precedenza:

```
export CLUSTER_NAME=ai-eks-docs
export AWS_REGION=us-east-2
```

Acquisisci il nome del bucket del modello Amazon S3 creato nelle fasi di configurazione del cluster. Il generatore di carico passa questo valore a vLLM.

```
MODEL_BUCKET=$(aws s3api list-buckets \
  --query "Buckets[?starts_with(Name, '${CLUSTER_NAME}-models-')].Name | [0]" \
  --output text)
echo "Model bucket: ${MODEL_BUCKET}"
```

Verificare che lo stack di monitoraggio sia in esecuzione

Lo stack kube-prometheus-stack della configurazione di monitoraggio viene eseguito nel namespace. `monitoring`

```
kubectl get pods -n monitoring
```

I pod Prometheus e Running Grafana dovrebbero essere:

```
NAME                                                     READY   STATUS    RESTARTS   AGE
kube-prometheus-stack-grafana-6f9c8b7d5c-2xk9p           3/3     Running   0          3h
kube-prometheus-stack-operator-7b8c9d6f4-q4m7n           1/1     Running   0          3h
prometheus-kube-prometheus-stack-prometheus-0            2/2     Running   0          3h
```

Se mancano, completa la configurazione del monitoraggio prima di continuare. ???

Verificare che il modello vLLM sia in esecuzione

I modelli vllm-inference-app Deployment and vllm-inference-svc Service from Load & Serve vengono eseguiti nel namespace. default

```
kubectl get deployment vllm-inference-app
```

Il rapporto di distribuzione dovrebbe essere pronto: 1/1

```
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
vllm-inference-app  1/1     1            1           3h
```

Se la distribuzione è mancante o non è pronta, completa i modelli Load & Serve prima di continuare.

Accedi a Grafana

Guarda il test di carico nella dashboard vLLM precaricata, quindi assicurati che il load balancer Grafana che hai impostato nella sezione Access Grafana sia raggiungibile. ??? Stampa il suo URL:

```
echo "http://$(kubectl get ingress kube-prometheus-stack-grafana -n monitoring -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')"
```

Apri l'URL nel tuo browser e accedi con nome utente admin e password con il seguente comando:

```
kubectl --namespace monitoring get secrets kube-prometheus-stack-grafana -o
 jsonpath="{.data.admin-password}" | base64 -d ; echo
```

Conferma che il vLLM ServiceMonitor esista

Il `vllm-inference-app` ServiceMonitor, creato nella fase Monitor vLLM, viene eseguito nel `default` namespace e indica a Prometheus dove raccogliere le metriche vLLM.

```
kubectl get servicemonitor vllm-inference-app
```

Output previsto:

```
NAME                AGE
vllm-inference-app   3h
```

Se manca, completa il passaggio Monitor vLLM. ???

Passaggio 1: riscaldare la GPU

Anche se il modello viene caricato nella memoria della GPU e la replica riporta Pronto, le prime richieste richiedono più tempo rispetto a quelle successive perché vLLM e la GPU terminano l'inizializzazione in base alle prime inferenze. vLLM acquisisce grafici CUDA, compila e ottimizza automaticamente i kernel della GPU e alloca i pool di memoria cache KV, mentre la GPU aumenta la velocità di clock da inattiva a quella di boost.

Si tratta di costi una tantum, quindi misurare le soglie prima di riscaldare i record di avvio funziona invece di servire allo stato stazionario. Il riscaldamento successivo invia 100 richieste sequenziali per completare l'inizializzazione, pertanto il test di carico che segue misura la capacità allo stato stazionario.

Per prima cosa, create il file ConfigMap che contiene lo script di riscaldamento (). `warmup.js` Lo script legge il modello di destinazione dalla variabile di `MODEL` ambiente, che Job imposta in base al `MODEL_BUCKET` valore acquisito in precedenza:

Example riscaldamento k6 ConfigMap

```
cat << 'EOF' | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: vllm-warmup-script
  namespace: default
```

```
data:
  warmup.js: |
    import http from 'k6/http';

    export default function () {
      const payload = JSON.stringify({
        model: __ENV.MODEL,
        prompt: 'Write a brief technical explanation about cloud computing concepts.',
        max_tokens: 100,
        temperature: 0.7,
      });
      const params = { headers: { 'Content-Type': 'application/json' }, timeout:
 '240s' };
      http.post('http://vllm-inference-svc:8000/v1/completions', payload, params);
    }
EOF
```

Quindi esegui il warm-up Job. Un singolo client invia 100 richieste sequenziali:

Example Warm-up Lavoro

```
cat << EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: vllm-warmup
  namespace: default
spec:
  backoffLimit: 0
  template:
    spec:
      restartPolicy: Never
      containers:
        - name: k6
          image: grafana/k6:latest
          env:
            - name: MODEL
              value: "s3://${MODEL_BUCKET}/Ministral-3-8B-Instruct-2512/"
          args: ["run", "--vus", "1", "--iterations", "100", "/scripts/warmup.js"]
          volumeMounts:
            - name: script
              mountPath: /scripts
      volumes:
        - name: script
```

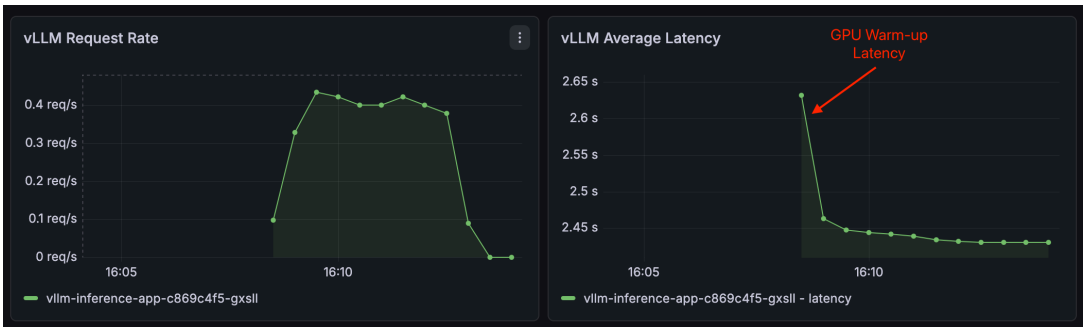
```
          configMap:
            name: vllm-warmup-script
EOF
```

Attendi che il riscaldamento finisca (ci vogliono alcuni minuti):

```
kubectl wait --for=condition=complete job/vllm-warmup --timeout=600s
```

Apri Grafana (apri il nome host del load balancer; vedi Access Grafana per i dettagli) e vai a Dashboards > GPU Monitoring > Performance Testing - VLLM Load Analysis. Il pannello VLM Request Rate mostra l'aumento del traffico di riscaldamento. Il pannello VLLM Average Latency mostra che la prima richiesta è più lenta, ossia il costo di riscaldamento una tantum della GPU.

Dashboard vLLM che mostra l'effetto di riscaldamento della GPU sulla latenza




Quindi elimina il job di riscaldamento:

```
kubectl delete job vllm-warmup
```

Passaggio 2: invia il carico a tassi di richiesta crescenti

In questo passaggio, si esegue un test di carico che invia richieste alla singola replica vLLM a velocità crescenti: 10, 20, 30, 40, 50, 60 e 70 richieste al secondo. Questo simula l'aumento della domanda da parte degli utenti. Il test utilizza k6 in un modello a tasso di arrivo costante, in cui la variabile di `TARGET_RPS` ambiente imposta la frequenza di richiesta target. Ogni frequenza viene eseguita per 60 secondi per consentire alle metriche di stabilizzarsi, seguita da una pausa di 30 secondi prima del successivo aumento della frequenza.

L'obiettivo di questo test è identificare il punto di saturazione, ovvero la frequenza di richiesta alla quale una singola replica non è più in grado di tenere il passo con le richieste in arrivo. È possibile monitorare la profondità della coda () e la latenza end-to-end (`vllm:num_requests_waiting`).

`vllm:e2e_request_latency_seconds` L'aumento della profondità e della latenza della coda indica che la replica si sta avvicinando alla saturazione.

Per prima cosa, create il file ConfigMap che contiene lo script k6 load-test (). `script.js` Lo script legge il modello di destinazione dalla variabile di `MODEL` ambiente, che ogni pod imposta in base al `MODEL_BUCKET` valore acquisito in precedenza:

Example test di carico k6 ConfigMap

```
cat << 'EOF' | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: vllm-loadtest-script
  namespace: default
data:
  script.js: |
    import http from 'k6/http';

    // Dictionary of 10-letter words for variation (prevents prompt caching)
    const words = [
      'revolution', 'technology', 'javascript', 'kubernetes', 'basketball',
      'strawberry', 'watermelon', 'blackberry', 'throughout', 'background',
      'playground', 'understand', 'everything', 'protection', 'complexity'
    ];

    export const options = {
      noConnectionReuse: true,
      scenarios: {
        constant_request_rate: {
          executor: 'constant-arrival-rate',
          rate: __ENV.TARGET_RPS || 10,  // Overridden by TARGET_RPS env var
          timeUnit: '1s',
          duration: __ENV.DURATION || '60s',  // Overridden by DURATION env var
          preAllocatedVUs: 50,
          maxVUs: 500,
        },
      },
    };

    export default function () {
      const randomWord = words[Math.floor(Math.random() * words.length)];
      const payload = JSON.stringify({
```

```
        model: __ENV.MODEL,
        prompt: `Starting with the word "${randomWord}", write a brief technical
 explanation about cloud computing concepts. Cover containerization, orchestration, and
 scalability in about 80 words.`,
        max_tokens: 100,
        temperature: 0.7,
      });
      const params = { headers: { 'Content-Type': 'application/json' }, timeout:
 '240s' };
      http.post('http://vllm-inference-svc:8000/v1/completions', payload, params);
    }
EOF
```

Quindi esegui il test di carico a ciascuna frequenza di richiesta. Il ciclo seguente invia un Job separato per ogni tariffa:

Example Loop di test di carico

```
for RPS in 10 20 30 40 50 60 70; do
  echo "=== Testing at ${RPS} req/s ==="
  cat << EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: vllm-loadtest-${RPS}
  namespace: default
  labels:
    app: vllm-loadtest
spec:
  backoffLimit: 0
  template:
    metadata:
      labels:
        app: vllm-loadtest
    spec:
      restartPolicy: Never
      containers:
        - name: k6
          image: grafana/k6:latest
          env:
            - name: K6_PROMETHEUS_RW_SERVER_URL
              value: "http://kube-prometheus-stack-
prometheus.monitoring.svc.cluster.local:9090/api/v1/write"
```

```
            - name: MODEL
              value: "s3://${MODEL_BUCKET}/Ministral-3-8B-Instruct-2512/"
            - name: TARGET_RPS
              value: "${RPS}"
            - name: DURATION
              value: "60s"
          args: ["run", "-o", "experimental-prometheus-rw", "/scripts/script.js"]
          volumeMounts:
            - name: script
              mountPath: /scripts
      volumes:
        - name: script
          configMap:
            name: vllm-loadtest-script
EOF
  kubectl wait --for=condition=complete job/vllm-loadtest-${RPS} --timeout=180s
  sleep 30
done
```

Guarda le metriche in Grafana mentre il test viene eseguito (vedi Passaggio 3). Ogni tasso crea il proprio Job (`vllm-loadtest-10`through`vllm-loadtest-70`), in modo da poter controllare ogni run con. `kubectl get jobs` Quando hai finito, elimina tutti i job di load-test:

```
kubectl delete jobs -l app=vllm-loadtest
```

Fase 3: Misura le metriche sotto carico

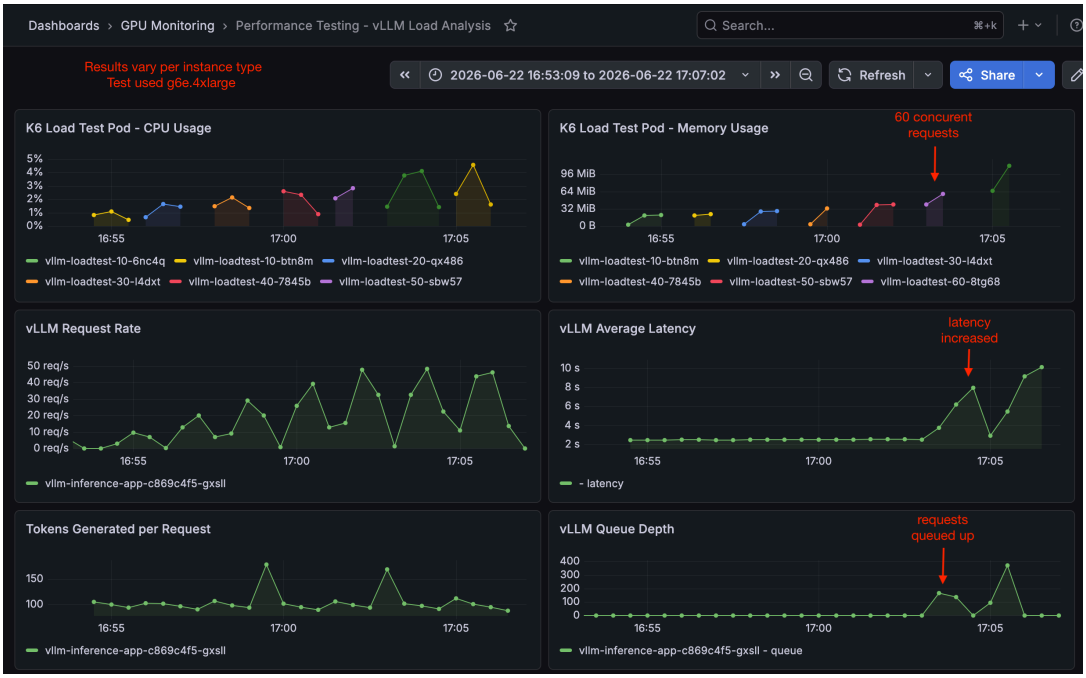
Mentre il carico viene eseguito, osserva le due metriche che determinano la scalabilità automatica:

- Profondità della coda (`vllm:num_requests_waiting`): quante richieste sono in attesa di essere elaborate.
- latenza end-to-end p95 (`vllm:e2e_request_latency_seconds`): il tempo di risposta del 95° percentile.

Visualizza le metriche in Grafana

Apri Grafana (apri il nome host del load balancer e accedi con nome`admin`; vedi Access Grafana per i dettagli) e vai a Dashboards > GPU Monitoring > Performance Testing - VLLM Load Analysis. Questa dashboard è basata su Amazon Managed Service for Prometheus, quindi i pannelli possono ritardare lo stato attivo fino a un minuto perché le metriche vengono scritte in remoto in batch.

Test delle prestazioni: dashboard di analisi del carico di vLLM durante il test di carico




In base ai risultati del test di carico, la profondità della coda è rimasta prossima allo zero su 50 richieste simultanee. È poi aumentata notevolmente con 60 richieste simultanee, con un picco compreso tra circa 150 e 350 richieste in coda. Allo stesso tempo, la latenza end-to-end di p95 è passata da una linea di base costante di circa 2,5 secondi a 6-10 secondi. Questo comportamento indica l'insorgenza di un sovraccarico prolungato e suggerisce che la scalabilità automatica dovrebbe iniziare prima che la profondità della coda raggiunga livelli che fanno sì che la latenza si avvicini ai 10 secondi. Un punto di partenza pratico è attivare il ridimensionamento quando la profondità della coda supera le 25 richieste per 30-60 secondi. Come segnale secondario, è possibile attivare il ridimensionamento anche quando la latenza end-to-end di p95 supera i 5 secondi nello stesso intervallo. Questo aiuta a tenere conto dei modelli di carico di lavoro in cui la latenza aumenta prima che la profondità della coda aumenti.

Fase 4: Scegli le metriche e le soglie su cui scalare

Utilizzando le metriche osservate nel punto di saturazione, determini le soglie utilizzate per configurare la scalabilità automatica nella sezione successiva.

- Profondità della coda (`vllm:num_requests_waiting`): il numero di richieste in attesa di essere elaborate una volta che una replica è saturata. Imposta la soglia al di sopra del punto in cui la coda rimane positiva per la prima volta, in modo che i picchi transitori non inneschino il ridimensionamento. Per il test di carico in questa sezione, la profondità della coda è rimasta prossima allo zero fino a 50 richieste simultanee ed è aumentata notevolmente fino a circa

150-350 richieste in coda su 60 richieste simultanee. Un pratico punto di partenza è attivare il ridimensionamento quando la profondità della coda supera le 25 richieste per 30-60 secondi.

- Latenza (`vllm:e2e_request_latency_seconds`): il tempo di risposta end-to-end quando una replica si avvicina al limite di capacità. Imposta la soglia al di sotto del tuo SLO di latenza, in modo che l'autoscaler si ridimensioni prima che la latenza visibile all'utente diventi inaccettabile. Per il test di carico in questa sezione, la latenza end-to-end di p95 è rimasta stabile a circa 2,5 secondi ed è aumentata a 6-10 secondi a saturazione. Come segnale secondario, un pratico punto di partenza è attivare il ridimensionamento quando la latenza end-to-end di p95 supera i 5 secondi per 30-60 secondi. Questo aiuta a tenere conto dei modelli di carico di lavoro in cui la latenza aumenta prima che la profondità della coda aumenti o quando la latenza aumenta senza code.
- Scale-down— la velocità con cui l'autoscaler rimuove le repliche dopo un calo della domanda. Riduci più lentamente di quanto aumenti. Per questo esempio, riduci la scala solo quando la profondità della coda è `0` e la latenza end-to-end di p95 è inferiore a 3 secondi per 5 minuti. Ciò impedisce all'autoscaler di rimuovere le repliche durante brevi pause nel traffico e aiuta a evitare le oscillazioni.

Questi valori forniscono un punto di partenza per la configurazione della scalabilità automatica nella sezione successiva. Ripeti questa procedura per il tuo modello, il tipo di GPU e i modelli di richiesta per determinare le soglie appropriate per la tua distribuzione.

## Inferenza AI con scalabilità automatica con HPA e KEDA

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Questa sezione mostra come scalare la `vllm-inference-app` distribuzione in base alle soglie stabilite nella sezione Trova le soglie delle metriche di scalabilità.

La procedura dettagliata utilizza i seguenti strumenti:

- KEDA (Kubernetes Event-Driven Autoscaler) è un progetto CNCF che interroga direttamente Prometheus e crea un Kubernetes Horizontal Pod Autoscaler (HPA) standard. Ridimensiona l'implementazione in base alla profondità della coda VLLM (il segnale di richiesta principale) e alla latenza end-to-end di p95 (un guardrail SLO), aggiungendo repliche quando una delle due soglie viene superata.

KEDA offre tre vantaggi principali per la scalabilità automatica dell'inferenza tramite GPU:

- Scalabilità a zero: KEDA è in grado di ridimensionare le implementazioni fino a zero repliche quando sono inattive e di ripristinarle quando la domanda ritorna, contribuendo a ridurre i costi delle GPU.
- Soglie di attivazione: KEDA separa la soglia che attiva la scalabilità dalla soglia target utilizzata per determinare il numero di repliche. Ciò consente di ignorare i picchi transitori ed evitare di riattivare le GPU per alcune richieste.
- Configurazione più semplice: per i carichi di lavoro di inferenza tramite GPU, KEDA è spesso il meccanismo di scalabilità automatica più semplice da configurare, con l'integrazione Prometheus integrata e nessun adattatore metrico separato da gestire.

Prerequisiti

Questa sottosezione continua da Find scaling Metric Thresholds. Assicurati di aver completato i modelli Load & Serve e la configurazione del monitoraggio, in modo che `vllm-inference-app` Deployment e Service siano in esecuzione nel `default` namespace e kube-prometheus-stack esegua lo scraping delle metriche VLLM.

Se hai aperto un nuovo terminale, imposta il nome e la regione del cluster che hai usato in precedenza:

```
export CLUSTER_NAME=ai-eks-docs
export AWS_REGION=us-east-2
```

Passaggio 1: installa KEDA

Installa KEDA with Helm nel suo namespace:

```
helm repo add kedacore https://kedacore.github.io/charts
helm repo update

helm install keda kedacore/keda --namespace keda --create-namespace
```

Verifica che i pod KEDA siano in funzione:

```
kubectl get pods -n keda
```

Output previsto:

```
NAME                                        READY   STATUS    RESTARTS   AGE
keda-admission-webhooks-7d4d6c6f9-xxxxx     1/1     Running   0          40s
keda-operator-6b8f9c5d7c-xxxxx              1/1     Running   0          40s
keda-operator-metrics-apiserver-xxxxx       1/1     Running   0          40s
```

Fase 2: Creare il KEDA ScaledObject

A `ScaledObject` indica a KEDA quale distribuzione scalare, i limiti della replica e i trigger. Questa configurazione utilizza due trigger: la profondità della coda come segnale di richiesta principale e la latenza end-to-end di p95 come guardrail SLO. L'`metricType: AverageValue`impostazione fa sì che KEDA divida ogni metrica tra le repliche, in modo che le soglie vengano interpretate per pod.

Questa procedura dettagliata utilizza le soglie di esempio della sezione Find scaling metric threshold: aumenta quando la profondità media della coda supera le 25 richieste in attesa per pod o quando la latenza end-to-end di p95 supera i 5 secondi. Sostituisci i valori misurati con il tuo modello, la GPU e le forme di richiesta.

```
cat << EOF | kubectl apply -f -
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: vllm-inference-app
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: vllm-inference-app
  minReplicaCount: 1
  maxReplicaCount: 5
  advanced:
    horizontalPodAutoscalerConfig:
      name: vllm-inference-app
      behavior:
        scaleUp:
          stabilizationWindowSeconds: 30
          policies:
            - type: Pods
              value: 2
              periodSeconds: 60
          selectPolicy: Max
        scaleDown:
          stabilizationWindowSeconds: 300
```

```
          policies:
            - type: Pods
              value: 1
              periodSeconds: 120
          selectPolicy: Min
  triggers:
    # Primary: queue depth (demand exceeds capacity)
    - type: prometheus
      metricType: AverageValue
      metadata:
        serverAddress: http://kube-prometheus-stack-prometheus.monitoring.svc:9090
        query: sum(vllm:num_requests_waiting) or vector(0)
        threshold: "25"
        activationThreshold: "1"
    # SLO guardrail: p95 end-to-end request latency (seconds)
    - type: prometheus
      metricType: AverageValue
      metadata:
        serverAddress: http://kube-prometheus-stack-prometheus.monitoring.svc:9090
        query: histogram_quantile(0.95,
 sum(rate(vllm:e2e_request_latency_seconds_bucket[1m])) by (le)) or vector(0)
        threshold: "5"
EOF
```

Le impostazioni principali:

- Trigger per la profondità della coda: `query` restituisce il numero totale di richieste in attesa su tutti i pod VLLm. La `or vector(0)` clausola restituisce `0` invece di un risultato vuoto quando nessuna richiesta è in attesa, il che impedisce al trigger di diventare inattivo. `threshold: "25"`è l'obiettivo di profondità di coda per pod stabilito in Find scaling Metric Thresholds e `activationThreshold: "1"` mantiene la distribuzione invariata `minReplicaCount` fino all'attesa di almeno una richiesta, quindi non si adatta a un segnale inattivo.
- Attivatore di latenza: `query` restituisce la latenza della richiesta end-to-end p95 in secondi, calcolata dall'istogramma di latenza vLLm in una finestra di 1 minuto. `threshold: "5"`è il guardrail SLO di latenza di Find scaling Metric Thresholds. Quando la latenza di p95 lo supera, KEDA aumenta la scalabilità anche se la coda non si è ancora creata.
- Valutazione dei trigger: KEDA è scalabile per soddisfare qualsiasi trigger che richieda il maggior numero di repliche, quindi una violazione di uno dei due segnali innesca uno scale-up.
- Scale-up e con un comportamento scalabile verso il basso: Scale-up è reattivo (finestra di 30 secondi, fino a 2 pod al minuto) perché in coda o in aumento della latenza significa che gli utenti

sono già in attesa. Scale-down è conservativo (finestra di 5 minuti, 1 pod ogni 2 minuti) perché i pod GPU si avviano lentamente. Questo approccio evita di rimuovere nuovamente la capacità che potrebbe essere necessaria pochi istanti dopo.

Verifica che ScaledObject sia pronto:

```
kubectl get scaledobject vllm-inference-app
```

Output previsto:

```
NAME                 SCALETARGETKIND      SCALETARGETNAME      MIN   MAX   READY
 ACTIVE
vllm-inference-app   apps/v1.Deployment   vllm-inference-app   1     5     True
 False
```

`READY: True`significa che i trigger sono validi e KEDA gestisce la distribuzione. `ACTIVE: False`è previsto quando non scorre traffico, poiché nessuna delle due soglie di attivazione è stata ancora superata.

Quando si crea il ScaledObject, KEDA crea e gestisce automaticamente un HPA (Horizontal Pod Autoscaler) di supporto con lo stesso nome. Non sei tu a creare o modificare questo HPA. Conferma che esista:

```
kubectl get hpa vllm-inference-app
```

Output previsto:

```
NAME                 REFERENCE                          TARGETS                MINPODS
 MAXPODS   REPLICAS
vllm-inference-app   Deployment/vllm-inference-app   0/25 (avg), 0/5 (avg)  1         5
         1
```

La `TARGETS` colonna mostra il valore corrente di ogni metrica rispetto alla relativa soglia (profondità della coda e latenza p95). `MINPODS`e `MAXPODS` provengono da `minReplicaCount` e `maxReplicaCount` dentro ScaledObject, ed `REPLICAS` è il numero attuale di pod VLLm.

Fase 3: Generazione del carico

Il test di carico riutilizza `vllm-loadtest-script` ConfigMap le soglie metriche di scalabilità create in Find. Conferma che esiste ancora:

```
kubectl get configmap vllm-loadtest-script
```

Se manca, ricrealo da Find scaling Metric Thresholds.

Esegui un singolo test di carico a 60 richieste al secondo per 10 minuti per favorire la scalabilità. Il Job legge il bucket del modello dalla variabile di `MODEL_BUCKET` ambiente impostata in precedenza e passa la frequenza e la durata attraverso `DURATION` le variabili `TARGET_RPS` e che lo script load-test legge.

```
cat << EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: vllm-loadtest-scaleup
  labels:
    app: vllm-loadtest-scaleup
spec:
  backoffLimit: 0
  template:
    spec:
      restartPolicy: Never
      containers:
        - name: k6
          image: grafana/k6:latest
          env:
            - name: MODEL
              value: "s3://${MODEL_BUCKET}/Ministral-3-8B-Instruct-2512/"
            - name: TARGET_RPS
              value: "60"
            - name: DURATION
              value: "10m"
          args: ["run", "/scripts/script.js"]
          volumeMounts:
            - name: script
              mountPath: /scripts
      volumes:
        - name: script
          configMap:
            name: vllm-loadtest-script
EOF
```

Il Job viene eseguito per 10 minuti. Passa alla fase successiva per osservare la scalabilità dell'implementazione durante l'esecuzione del test.

Fase 4: Guarda la scalabilità dell'implementazione

Osserva la risposta dell'autoscaler al carico del passaggio precedente. Esegui ciascuno dei seguenti comandi in un terminale separato.

Guarda l'HPA. La `TARGETS` colonna supera gli obiettivi configurati e `REPLICAS` aumenta:

```
kubectl get hpa vllm-inference-app -w
```

Guarda i pod VLLm. Le nuove repliche appaiono e vengono visualizzate `Ready` una volta che il modello viene caricato nella memoria della GPU:

```
kubectl get pods -l app=vllm-inference-app -w
```

Conferma che ScaledObject è attivo. `ACTIVE`passa a `True` una volta superata una soglia di attivazione:

```
kubectl get scaledobject vllm-inference-app
```

Comportamento previsto:

- Entro circa 30 secondi la profondità della coda supera la soglia (o la latenza end-to-end di p95 supera i 5 secondi) e la colonna `TARGETS` HPA supera gli obiettivi configurati.
- KEDA amplia l'implementazione e Karpenter fornisce un nuovo nodo GPU se non è disponibile. Le nuove repliche arrivano `Ready` una volta che il modello viene caricato nella memoria della GPU.
- Man mano che il traffico si distribuisce tra le repliche, la coda si esaurisce, la latenza si ripristina ed entrambe le metriche tornano al di sotto delle rispettive soglie.
- Dopo l'arresto del carico, l'implementazione mantiene il numero di repliche più elevato fino al tempo di recupero di 5 minuti, quindi viene ridimensionato di un pod alla volta. `minReplicaCount`

Al termine del test di carico, eliminate il load-test Job. Lascialo al suo `vllm-loadtest-script` ConfigMap posto, poiché è condiviso con la sezione delle soglie:

```
kubectl delete jobs -l app=vllm-loadtest-scaleup --ignore-not-found
```

Eliminazione

> **Note**
>
> Se prevedi di continuare a utilizzare la scalabilità automatica, salta questa pulizia. Eseguilo solo quando hai finito.

Per rimuovere le risorse di scalabilità automatica che hai creato in questa sezione, elimina `ScaledObject` e disinstalla KEDA:

```
kubectl delete scaledobject vllm-inference-app --ignore-not-found
helm uninstall keda -n keda
kubectl delete namespace keda
```

L'eliminazione di rimuove `ScaledObject` anche l'HPA creato da KEDA. L'implementazione VLLm torna al numero di repliche statiche.

Per rimuovere il server di inferenza VLLm e le relative risorse del carico di lavoro, consulta Load & Serve Models. Per istruzioni sulla rimozione delle risorse dell'infrastruttura come il cluster e il bucket S3 NodePool, consulta Cluster Setup Cleanup.

## Accelera il caricamento dei modelli su Amazon EKS

Quando distribuisci modelli linguistici di grandi dimensioni (LLM) su Amazon EKS, il tempo di caricamento dei modelli influisce direttamente sulla rapidità con cui i Pods possono iniziare a fornire richieste di inferenza. Ciò è particolarmente vero durante gli eventi di scalabilità, quando nuovi pod o nodi devono caricare il modello prima di gestire il traffico. L'avvio del modello prevede due fasi che puoi migliorare con l'ottimizzazione e la memorizzazione nella cache degli artefatti:

- Caricamento dei pesi: streaming dei file di peso del modello da Amazon S3 nella memoria della GPU utilizzando Model Streamer. Run:ai
- torch.compile — Compilazione del grafico di calcolo del modello in kernel fused ottimizzati. CUDA/ Triton Questa compilazione viene eseguita al primo avvio e può influire in modo significativo sull'ora di avvio a seconda delle dimensioni del modello.

Questo argomento mostra come ottimizzare le prestazioni di Run:ai Model Streamer e il caching di torch.compile per ridurre entrambe le fasi del processo di caricamento del modello. Per la procedura

completa di distribuzione di vLLM su Amazon EKS per l'inferenza, consulta. the section called "Modello Load & Serve"

## Ottimizza il percorso di rete S3 in EKS Auto Mode

Se utilizzi la modalità EKS Auto con nodi GPU in sottoreti private, ti consigliamo di utilizzare un endpoint Gateway VPC per S3 per ottimizzare il percorso di rete tra i tuoi nodi e S3. Con l'endpoint Gateway VPC, il traffico verso S3 rimane sulla AWS rete e bypassa completamente il NAT Gateway, quindi non vi è alcun limite di larghezza di banda condiviso e nessun costo di elaborazione dei dati NAT per GB. Senza l'endpoint Gateway VPC, quando il traffico attraversa un NAT Gateway, il NAT Gateway diventa un collo di bottiglia condiviso durante la scalabilità, quando più nodi eseguono il modello contemporaneamente.

La modalità automatica EKS colloca comunemente i nodi in sottoreti private, quindi il traffico verso S3 fluisce attraverso un gateway NAT per impostazione predefinita. Un gateway NAT fornisce fino a 100 Gbps di larghezza di banda e 55.000 connessioni simultanee per destinazione. Tuttavia, tale larghezza di banda è condivisa tra tutti i nodi della sottorete privata. Durante un evento di scalabilità, diversi nodi che scaricano contemporaneamente il modello completo si contendono la stessa larghezza di banda del NAT Gateway, il che può rallentare il caricamento dei pesi del modello su tutti.

## Run:ai Ottimizzazione delle prestazioni di Model Streamer

I motori di inferenza come vLLM e SGlang utilizzano Run:ai Model Streamer come meccanismo alternativo per caricare i pesi durante l'avvio dell'inferenza.

Per impostazione predefinita, Run:ai Model Streamer utilizza impostazioni di concorrenza conservative quando scarica i file di peso del modello da S3. L'aumento della concorrenza di download e della dimensione dei blocchi riduce il tempo di caricamento dei pesi del modello scaricando più dati in parallelo.

Calcola la concorrenza ottimale

Calcola il valore di concorrenza ottimale nel modo seguente:

```
concurrency = ceil(total_model_size_gb / chunk_size_gb)
```

Sostituisci il `concurrency` valore che usi in base alle dimensioni del modello e alla dimensione del blocco. Per alcuni esempi, consultate la tabella seguente. Ad esempio, con un modello da 67 GB e una dimensione del blocco da 4 GB:`ceil(67 / 4) = 17`.

| Dimensione del modello | Dimensione del blocco | Concurrency (Simultaneità) |
|---|---|---|
| 10 GB | 4 GB | 3 |
| 67 GB | 4 GB | 17 |
| 140 GB | 4 GB | 35 |

Applica la configurazione

Aggiungi i seguenti argomenti e variabili di ambiente alle specifiche del contenitore di inferenza:

- `--tensor-parallel-size`— Il grado tensoriale parallelo (TP) è in genere il numero minimo di GPU richiesto per adattare il modello alla memoria della GPU in base alle dimensioni del modello. Ad esempio, un modello da 67 GB su un tipo di `p5.48xlarge` istanza richiede almeno 2 GPU, quindi impostalo su. 2
- `concurrency`and `distributed` (in`--model-loader-extra-config`): impostato `concurrency` sul valore calcolato per il modello. Impostato `distributed` su `true` solo quando si utilizza il parallelismo tensoriale (TP > 1). Se abilitato, ogni rango tensore-parallelo trasmette il proprio frammento di peso direttamente da Amazon S3, invece di caricare tutti i pesi al rango 0 e trasmetterlo agli altri ranghi. Ciò migliora notevolmente le prestazioni di caricamento per le distribuzioni con più GPU. Lascialo deimpostato (`ofalse`) per TP=1, dove non fornisce alcun vantaggio. Questa opzione richiede l'architettura vLLM V1. È incompatibile con`--enforce-eager`, che impone il percorso V0; utilizzandoli entrambi insieme si genererà un errore o si tornerà automaticamente al caricamento non distribuito.
- `RUNAI_STREAMER_CHUNK_BYTESIZE`— Dimensione del blocco da 4 GB. Questo valore mostra costantemente le migliori prestazioni tra i benchmark. Blocchi più grandi riducono il numero di richieste S3 e migliorano il throughput su istanze con larghezza di banda elevata.
- `RUNAI_STREAMER_S3_REQUEST_TIMEOUT_MS`— Per-request timeout in millisecondi. Consente un nuovo tentativo più rapido sulle risposte S3 lente.
- `RUNAI_STREAMER_S3_LOW_SPEED_LIMIT`— Velocità minima di trasferimento in byte al secondo prima che una richiesta venga considerata lenta e ritentata.

```
containers:
```

```
- name: vllm-inference
  image: vllm/vllm-openai:v0.21.0
  command:
    - python3
    - -m
    - vllm.entrypoints.openai.api_server
  args:
  # ... your existing args ...
  - --model=s3://<MODEL_PATH>
  - --tensor-parallel-size=2
  - --load-format=runai_streamer
  - --model-loader-extra-config={"concurrency":17,"distributed":true}
  env:
  - name: RUNAI_STREAMER_CHUNK_BYTESIZE
    value: "4294967296"
  - name: RUNAI_STREAMER_S3_REQUEST_TIMEOUT_MS
    value: "3000"
  - name: RUNAI_STREAMER_S3_LOW_SPEED_LIMIT
    value: "1048576"
```

## Ottimizza torch.compile cold start

La `torch.compile` fase del processo di elaborazione delle inferenze traccia il grafico di calcolo di un modello (la sequenza di operazioni matematiche) e lo compila in kernel fused ottimizzati. CUDA/ Triton Non compila i pesi del modello, ma solo le operazioni che li trasformano.

I motori di inferenza utilizzano torch.compile perché fornisce automaticamente significativi miglioramenti del throughput, senza ingegneria del kernel personalizzata per architettura del modello:

- Fusione del kernel: diverse piccole operazioni (aggiunta residua, layernorm, attivazione) vengono fuse in un unico kernel, riducendo i roundtrip della memoria della GPU.
- Meno avvii del kernel: un trasformatore passa da circa 15-30 kernel CUDA separati a circa 10 kernel fusi, risparmiando il sovraccarico della CPU per avvio.
- Python rimosso dall'hot path: l'intero forward pass diventa un piano di esecuzione C++, eliminando il sovraccarico dell'interprete Python tra le operazioni.
- Migliore compatibilità con CUDA Graph: i grafici statici compilati vengono acquisiti e riprodotti con un sovraccarico della CPU quasi nullo.
- Ottimizzazione  automatica: funziona per qualsiasi architettura di modello. La produttività migliora del 5-30% rispetto alla modalità eager.

La tabella seguente mostra i motori di inferenza più comuni che utilizzano torch.compile.

| Motore | utilizzo di torch.compile |
|---|---|
| vLLM | Abilitato per impostazione predefinita (architettura V1) |
| SGLang | Facoltativo tramite `--enable-torch-compile` |
| TensorRT-LLM | Un nuovo percorso lo supporta insieme alla creazione del motore precedente |

Il problema dell'avvio a freddo di torch.compile

Il compromesso di torch.compile è che la prima inferenza Pod deve essere compilata prima di poter soddisfare le richieste. Questa compilazione può richiedere fino a diversi minuti a seconda delle dimensioni del modello. Gli artefatti compilati sono piccoli (~15 MB per un modello da 60 GB) ma aumentano notevolmente il tempo di avvio a freddo. Poiché gli artefatti sono piccoli e deterministici per una determinata configurazione, è possibile memorizzarli nella cache e riutilizzarli per eliminare la penalizzazione dovuta all'avvio a freddo al successivo avvio del Pod e del nodo.

Gli artefatti sono costituiti da:

- File sorgente del kernel Triton generati
- Binari del kernel compilati () `.cubin`
- Struttura grafica che sequenzia le chiamate al kernel

Il compromesso --enforce-eager

Per impostazione predefinita, vLLM abilita l'acquisizione di grafici CUDA. `torch.compile` Il `--enforce-eager` flag disattiva entrambi ed esegue il modello in modalità eager, in cui ogni operazione viene eseguita immediatamente tramite l'interprete Python. Poiché la modalità eager salta sia la compilazione che l'acquisizione di grafici, alcune guide rapide, inclusa la distribuzione di base in, vengono utilizzate per avviare i Pod più velocemente. the section called “Modello Load & Serve” `--enforce-eager`

`--enforce-eager`è una scelta valida per il debug, le implementazioni con limiti di memoria o le architetture di modelli che non vengono compilate in modo corretto, ed è possibile utilizzarla in produzione per questi motivi. Sebbene possa mitigare i problemi legati all'avvio a `torch.compile`

freddo, consigliamo altri approcci, come quelli descritti nelle sezioni successive di questa pagina, per mantenere le prestazioni di runtime in produzione.

Comprendi il compromesso tra tempo di avvio e prestazioni in fase di esecuzione prima dell'implementazione in produzione. `--enforce-eager`

| Aspetto | Con (modalità eager) **`--enforce-eager`** | Predefinito (torch.compile + grafici CUDA) |
|---|---|---|
| Ora di avvio | Veloce: nessuna fase di compilazione o acquisizione di grafici | Avvio lento a freddo (problema risolto con the section called "Memorizza gli artefatti torch.compile sullo stesso nodo" ethe section called "Pre-warm torch.compile cache sui nuovi nodi") |
| Steady-state produttività | Linea di base | ~5-30% in più rispetto alla fusione del kernel |
| Per-token latenza (piccoli lotti) | Maggiore sovraccarico di avvio della CPU | Molto più basso: il kernel di riproduzione dei grafici CUDA viene avviato come un'unica unità |
| Memoria GPU | Più basso e più prevedibile | Più alto: l'acquisizione del grafico prealloca i pool di buffer |
| Debuggabilità | Pulisci le tracce dello stack per operazione | Gli errori emergono all'interno dei kernel generati |

## Memorizza gli artefatti torch.compile sullo stesso nodo

Questa tecnica si applica quando si utilizza un motore di inferenza che supporta`torch.compile`, ad esempio vLLM (abilitato per impostazione predefinita) o SGlang (abilitato tramite). `--enable-torch-compile` Funziona solo quando `torch.compile` è attivo, cioè quando non è impostato. `--enforce-eager`

> **Important**
>
> Questo miglioramento non ha effetto se `torch.compile` è disabilitato. In vLLM, il `--enforce-eager` flag si disattiva `torch.compile` completamente, quindi nessun artefatto viene compilato o memorizzato nella cache. Se hai seguito la distribuzione di base the section called "Modello Load & Serve" con`--enforce-eager`, vLLM crea la directory della cache ma non vi scrive mai. Rimuovi `--enforce-eager` prima di applicare questa tecnica.

Quando un motore di inferenza compila il grafico di calcolo del modello al primo avvio, puoi memorizzare nella cache i kernel ottimizzati risultanti nella memoria locale del nodo. I pod successivi sullo stesso nodo riutilizzano gli artefatti memorizzati nella cache e saltano completamente la fase di compilazione, il che può ridurre significativamente i tempi di avvio.

Aggiungi variabili di ambiente alla cache

Aggiungete le seguenti variabili di ambiente alle specifiche del contenitore di inferenza per indirizzare torch.compile e Triton cache a un percorso host persistente. Ti consigliamo di utilizzare lo store di istanze NVMe locale del nodo e non il volume principale di Amazon Elastic Block Store (Amazon EBS) per. `hostPath` Per esempi, consulta la sezione seguente.

```
containers:
- name: vllm-inference
  env:
  # torch.compile cache
  - name: XDG_CACHE_HOME
    value: "/compile-cache"
  - name: TORCHINDUCTOR_CACHE_DIR
    value: "/compile-cache/inductor"
  - name: TRITON_CACHE_DIR
    value: "/compile-cache/triton"
  volumeMounts:
  - name: compile-cache
    mountPath: /compile-cache
volumes:
- name: compile-cache
  hostPath:
    path: /mnt/k8s-disks/0/compile-cache
    type: DirectoryOrCreate
```

Imposta il percorso della cache per l'archivio di istanze NVMe

Gli artefatti compilati e i pesi trasmessi in streaming traggono vantaggio dall'archiviazione locale veloce. Nelle istanze GPU con storage di istanze NVMe (come le P-family istanze G-family and), l'instant store ne fornisce circa 30 GB/s, rispetto a circa 1 del volume Amazon EBS principale. GB/s Indirizza la cache al punto di montaggio `hostPath` NVMe per ottimizzare il throughput.

> **Important**
>
> Il `hostPath` volume utilizza. `type: DirectoryOrCreate` Se lo indirizzi verso un percorso non supportato dall'Instance Store NVMe, Kubernetes crea invece silenziosamente la directory sul volume principale di Amazon EBS. La cache funziona ancora, ma si perdono i vantaggi prestazionali di NVMe senza errori o avvisi.

Il punto di montaggio NVMe e il modo in cui viene abilitato differiscono tra la modalità automatica EKS e i nodi autogestiti:

| Calcolo | Punto di montaggio NVMe | Come abilitare l'archivio di istanze NVMe |
|---|---|---|
| Modalità automatica di EKS | `/mnt/.ephemeral` | Abilitato dinamicamente in base allo storage temporaneo richiesto. EKS Auto Mode formatta e monta l'archivio di istanze NVMe come array RAID 0 quando l'istanza ha più unità NVMe solo quando la `ephemeralStorage.size` richiesta NodeClass è inferiore alla capacità NVMe disponibile dell'istanza. Se la richiesta `ephemeralStorage.size` è uguale o superiore alla capacità NVMe, EKS Auto Mode non utilizza l'archivio di istanze e il percorso è invece supportato dal volume EBS principale. |

| Calcolo | Punto di montaggio NVMe | Come abilitare l'archivio di istanze NVMe |
|---|---|---|
| Self-managed Karpenter | `/mnt/k8s-disks/0` | Ambientato `instanceStorePolicy: RAID0` nel Karpenter. `EC2NodeClass` Senza di esso, Karpenter ignora i volumi dell'instance-store e il percorso non è supportato da NVMe. |

Per EKS Auto Mode, imposta il valore su nelle specifiche del contenitore: `hostPath` `/mnt/.ephemeral/compile-cache`

```
volumes:
- name: compile-cache
  hostPath:
    path: /mnt/.ephemeral/compile-cache
    type: DirectoryOrCreate
```

Per Karpenter autogestito, imposta il `hostPath` to `/mnt/k8s-disks/0/compile-cache` nelle specifiche del contenitore:

```
volumes:
- name: compile-cache
  hostPath:
    path: /mnt/k8s-disks/0/compile-cache
    type: DirectoryOrCreate
```

Risultati di esempio

1. Il primo Pod su un nodo esegue torch.compile e scrive i kernel compilati sull'host. `/compile-cache`
2. I Pod successivi sullo stesso nodo montano la cache esistente e saltano completamente la compilazione, il che riduce l'avvio a freddo di torch.compile da ~50—80 s a ~4—6 s.

La tabella seguente mostra il miglioramento della memorizzazione nella cache dello stesso nodo per modelli di dimensioni diverse:

| Dimensione del modello | first Pod torch.compile (senza cache) | Pods torch.compile successivi (stesso nodo) |
|---|---|---|
| 60 GB | ~ 53 secondi | ~6 secondi |
| 140 GB | ~ anni '60 | ~6 secondi |
| 640 GB | ~anni '80 | ~6 secondi |

### Pre-warm torch.compile cache sui nuovi nodi

Questa tecnica si applica quando si esegue l'inferenza multinodo con GPU omogenee, parallelismo tensoriale, modelli e versioni tra nodi. PyTorch La tecnica the section called "Memorizza gli artefatti torch.compile sullo stesso nodo" si concentra sul caso di un singolo nodo, ma i nuovi nodi aggiunti durante gli eventi di scalabilità iniziano con una cache torch.compile vuota. Per ridurre ulteriormente i tempi di avvio a freddo sui nodi appena inizializzati, puoi implementare un meccanismo di caching che memorizza gli artefatti torch.compile compilati in S3 e li prescarica in nuovi nodi quando si uniscono al cluster.

L'approccio generale è:

1. Dopo che il primo Pod ha compilato il modello sul primo nodo, carica gli artefatti torch.compile (~15 MB) in un bucket S3.
2. Quando nuovi nodi si uniscono al cluster, scarica gli artefatti memorizzati nella cache nella memoria locale del nodo prima che i Pod di inferenza siano programmati.

Ad esempio, puoi implementare uno DaemonSet che viene eseguito su nodi GPU che impacchetta e carica la cache torch.compile su S3. Può anche sincronizzare la cache con l'archiviazione dei nodi locali prima che i pod di inferenza vengano pianificati su nuovi nodi.

Considerazioni sulla memorizzazione nella cache tra nodi

Quando si memorizzano nella cache gli artefatti torch.compile tra i nodi, i kernel compilati sono validi solo quando questi parametri corrispondono tra il nodo che ha generato la cache e il nodo che la consuma. Il meccanismo di caching deve tenere conto di tutti questi fattori. Una mancata corrispondenza su qualsiasi parametro produce una cache non valida che impone la ricompilazione o causa errori di runtime. Lo strumento di caching deve distinguere gli artefatti in base a questi

parametri, ad esempio incorporandoli nella chiave oggetto S3 o nella struttura della directory della cache.

| Parametro | Perché è importante |
|---|---|
| Tipo GPU | I kernel compilati sono GPU-architecture-specific (ad esempio, sm_90 per H100 e sm_89 per L4). |
| Parallelismo tensoriale (TP) | Gradi TP diversi producono partizioni grafiche di calcolo diverse. |
| Modello | L'architettura e la dimensione di ogni modello vengono compilate in kernel diversi. |
| PyTorch versione | Gli interni dei compilatori torch.compile e Triton possono introdurre modifiche sostanziali tra le versioni. |

Risultati di esempio

Nei test direzionali con preriscaldamento della cache tra nodi (Qwen3-6-35B-A3B, 67 GB, 2 GPU con TP=2 su p5.48xlarge), il primo Pod su nodi appena scalati ha raggiunto lo stesso tempo di avvio dei Pod successivi su un nodo già caldo:

| Scenario | Primo Pod | Secondo pod |
|---|---|---|
| Senza memorizzazione nella cache tra nodi (nuovo nodo) | 65s | Anni 16 |
| Con memorizzazione nella cache tra nodi (nuovo nodo) | 16s | 16 anni |

## Esempio di implementazione

L'esempio seguente combina l'ottimizzazione delle prestazioni di Run:ai Model Streamer e una cache torch.compile in un unico manifest di vLLM Deployment. Sostituisci i valori segnaposto con la tua configurazione:

- `serviceAccountName`— Account di servizio con un ruolo IAM che consente ad Amazon S3 di accedere in lettura al bucket del modello.
- `nodeSelector`(`karpenter.sh/nodepool`) — Il nome del pool di nodi della GPU (ad esempio,). `gpu-nodepool-g6e-12xlarge`
- `--model`— Percorso Amazon S3 rispetto ai pesi del modello.
- `--model-loader-extra-config`— Impostato `concurrency` in base alle dimensioni del modello:. `ceil(total_model_size_gb / chunk_size_gb)` Ad esempio, un modello da 67 GB con una dimensione del blocco da 4 GB fornisce`ceil(67 / 4) = 17`.
- `--tensor-parallel-size`— Impostare il grado tensoriale parallelo (TP) sul numero minimo di GPU necessarie per adattare il modello alla memoria.
- `hostPathpath`— Mount point dell'archivio di istanze NVMe per nodi autogestiti con Karpenter. In modalità EKS Auto, usa invece. `/mnt/.ephemeral/compile-cache` Per informazioni dettagliate, vedi the section called "Memorizza gli artefatti torch.compile sullo stesso nodo".

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vllm-inference
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vllm-inference
  template:
    metadata:
      labels:
        app: vllm-inference
    spec:
      serviceAccountName: <SERVICE_ACCOUNT_NAME>
      nodeSelector:
        karpenter.sh/nodepool: <GPU_NODEPOOL>
      tolerations:
        - key: nvidia.com/gpu
          operator: Exists
          effect: NoSchedule
      containers:
        - name: vllm
          image: vllm/vllm-openai:v0.21.0
```

```
          command:
            - python3
            - -m
            - vllm.entrypoints.openai.api_server
          args:
            - --model=s3://<BUCKET_NAME>/<MODEL_PATH>
            - --load-format=runai_streamer
            - --model-loader-extra-config={"concurrency":17,"distributed":true}
            - --tensor-parallel-size=2
            - --max-model-len=8192
            - --host=0.0.0.0
            - --port=8000
          ports:
            - containerPort: 8000
              name: http
          env:
            # Run:ai streamer tuning
            - name: RUNAI_STREAMER_CHUNK_BYTESIZE
              value: "4294967296"
            - name: RUNAI_STREAMER_S3_REQUEST_TIMEOUT_MS
              value: "3000"
            - name: RUNAI_STREAMER_S3_LOW_SPEED_LIMIT
              value: "1048576"
            # torch.compile cache
            - name: XDG_CACHE_HOME
              value: "/compile-cache"
            - name: TORCHINDUCTOR_CACHE_DIR
              value: "/compile-cache/inductor"
            - name: TRITON_CACHE_DIR
              value: "/compile-cache/triton"
          resources:
            requests:
              cpu: "12"
              memory: 80Gi
              nvidia.com/gpu: "2"
            limits:
              nvidia.com/gpu: "2"
          volumeMounts:
            - name: compile-cache
              mountPath: /compile-cache
          startupProbe:
            httpGet:
              path: /health
              port: 8000
```

```
            periodSeconds: 10
            failureThreshold: 60
            initialDelaySeconds: 30
          readinessProbe:
            httpGet:
              path: /health
              port: 8000
            periodSeconds: 5
            timeoutSeconds: 3
      volumes:
        - name: compile-cache
          hostPath:
            path: /mnt/k8s-disks/0/compile-cache
            type: DirectoryOrCreate
```

# Configurazione del cluster Amazon EKS per AI/ML carichi di lavoro

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Questa sezione è progettata per aiutarti a configurare i cluster Amazon EKS ottimizzati per i AI/ML carichi di lavoro. Troverai indicazioni sull'esecuzione di GPU-accelerated container utilizzando AMI ottimizzate per Linux e Windows, sulla configurazione di cluster di addestramento con Elastic Fabric Adapter (EFA) per reti ad alte prestazioni e sulla creazione di cluster di inferenza AWS con istanze Inferentia, inclusi prerequisiti, procedure dettagliate e considerazioni sulla distribuzione.

Argomenti

- Usa le AMI EKS-optimized accelerate per le istanze GPU
- Esegui GPU-accelerated contenitori (Windows su EC2) G-Series
- Esecuzione dei corsi di machine learning su Amazon EKS con Elastic Fabric Adapter
- Utilizzo AWS Istanze Inferentia con Amazon EKS per l'apprendimento automatico

## Usa le AMI EKS-optimized accelerate per le istanze GPU

Amazon EKS supporta le AMI EKS-optimized Amazon Linux e Bottlerocket per le istanze GPU. Le AMI EKS-optimized accelerate semplificano l'esecuzione dei carichi di lavoro AI e ML nei cluster EKS

fornendo immagini del sistema operativo predefinite e convalidate per lo stack Kubernetes accelerato. Oltre ai componenti principali di Kubernetes inclusi nelle EKS-optimized AMI standard, le AMI EKS-optimized accelerate includono i moduli e i driver del kernel necessari per eseguire le istanze NVIDIA GPU G ed P EC2 e le istanze AWS GPU Inferentia e Trainium EC2 nei cluster EKS.

La tabella seguente mostra i tipi di istanze GPU supportati per ciascuna variante di AMI accelerata. EKS-optimized Consulta le versioni EKS-optimized AL2023 e Bottlerocket GitHub per gli ultimi aggiornamenti alle varianti AMI.

| Variante EKS AMI | Tipi di istanza EC2 |
|---|---|
| NVIDIA AL 2023 x86_64 | p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, g7e, gr6, g6, g6e, g6f, gr6f, gr6f, g5, g4dn |
| AL2023 PER NVIDIA | p6e-gb200, p6e-gb300, g5g |
| Neurone AL2023 x86_64 | inf1, inf2, trn1, trn2 |
| Bottlerocket x86_64 aws-k8s-nvidia | p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, g7e, gr6, g6, g6e, g6f, gr6f, g5, g4dn |
| aarch64/arm64 Bottlerocket aws-k8s-nvidia | 5 g |
| Bottlerocket x86_64 aws-k8s | inf1, inf2, trn1, trn2 |

**Important**

Il tipo di istanza G7 EC2 richiede il driver NVIDIA versione 595 o successiva. Le AMI EKS-optimized accelerate attualmente includono il driver NVIDIA versione 580, che non supporta le istanze G7. Per utilizzare le istanze G7 con Amazon EKS, devi creare un'AMI personalizzata con il driver NVIDIA versione 595 utilizzando gli script di compilazione EKS AMI. ???

Se utilizzi Karpenter per il provisioning dei nodi e il ridimensionamento automatico con EKS, si consiglia di escludere la famiglia di istanze dalla selezione automatica dell'AMI che utilizza la g7 famiglia di istanze. NodePools Consulta `amiSelectorTerms` la documentazione di

Karpenter per sapere come configurare Karpenter per utilizzare l'AMI personalizzata per le istanze NodeClass . g7

## EKS-optimized AMI NVIDIA

Utilizzando le AMI EKS-optimized NVIDIA, l'utente accetta il Cloud End User License Agreement (EULA) di NVIDIA.

Per trovare le AMI EKS-optimized NVIDIA più recenti, consulta e. the section called “Ottieni le ultime novità IDs” the section called “Ottenimento degli ID della versione più recente”

Quando usi Amazon Elastic Fabric Adapter (EFA) con le AMI NVIDIA EKS-optimized AL2023 o Bottlerocket, devi installare il plug-in del dispositivo EFA separatamente. Per ulteriori informazioni, consulta the section called “Configurazione dei cluster di formazione con EFA”.

## AMI NVIDIA EKS AL2023

Quando si utilizza l'operatore GPU NVIDIA con le AMI NVIDIA EKS-optimized AL2023, è necessario disabilitare l'installazione da parte dell'operatore del driver e del toolkit, poiché questi sono già inclusi nelle AMI EKS. Le AMI NVIDIA EKS-optimized AL2023 non includono il plug-in del dispositivo NVIDIA Kubernetes o il driver NVIDIA DRA e devono essere installate separatamente. Per ulteriori informazioni, consulta the section called “GPU NVIDIA”.

Oltre ai componenti EKS AMI standard, le AMI NVIDIA AL2023 includono i seguenti componenti. EKS-optimized

- Driver NVIDIA
- Driver in modalità utente NVIDIA CUDA
- Toolkit per contenitori NVIDIA
- NVIDIA fabric manager
- NVIDIA ha persistito
- Driver NVIDIA IMEX
- NVIDIA NVLink Subnet Manager
- Driver NVIDIA GDRCopy
- EFA minimal (modulo kernel e rdma-core)

Per dettagli sul driver in modalità utente NVIDIA CUDA e sul CUDA runtime/libraries utilizzato nei contenitori delle applicazioni, consultate la documentazione di NVIDIA. https://docs.nvidia.com/deploy/cuda-compatibility/why-cuda-compatibility.html#why-cuda-compatibility La versione CUDA mostrata `nvidia-smi` è la versione del driver in modalità utente NVIDIA CUDA installato sull'host, che deve essere compatibile con il CUDA utilizzato nei contenitori di applicazioni. runtime/libraries

Le AMI NVIDIA EKS-optimized AL2023 supportano il kernel 6.12 per le versioni di Kubernetes 1.33 e successive e la versione del driver NVIDIA 580 per tutte le versioni di Kubernetes. Il driver NVIDIA 580 è necessario per utilizzare CUDA 13+.

Consulta le versioni EKS-optimized AL2023 GitHub per i dettagli sulle versioni dei componenti incluse nelle AMI. Consulta lo script di installazione dell'AMI NVIDIA EKS AL2023 e lo script di caricamento del kernel per dettagli su come le AMI EKS configurano le dipendenze NVIDIA. Puoi trovare l'elenco dei pacchetti installati e le relative versioni su un'istanza EC2 in esecuzione con il comando. `dnf list installed`

Quando si creano AMI personalizzate con le EKS-optimized AMI come base, non è consigliabile o supportato eseguire un aggiornamento del sistema operativo (ad esempio`dnf upgrade`) o aggiornare uno qualsiasi dei pacchetti Kubernetes o GPU inclusi nelle EKS-optimized AMI, poiché ciò rischia di compromettere la compatibilità dei componenti. Se si aggiorna il sistema operativo o i pacchetti inclusi nelle EKS-optimized AMI, si consiglia di eseguire dei test approfonditi in un ambiente di sviluppo o di staging prima dell'implementazione in produzione.

Quando si creano AMI personalizzate per istanze GPU, si consiglia di creare AMI personalizzate separate per ogni generazione e famiglia di tipi di istanze da eseguire. Le AMI EKS-optimized accelerate installano selettivamente driver e pacchetti in fase di esecuzione in base alla generazione e alla famiglia del tipo di istanza sottostante. Per ulteriori informazioni, consulta gli script EKS AMI per l'installazione e il runtime. https://github.com/awslabs/amazon-eks-ami/blob/main/templates/al2023/runtime/gpu/nvidia-kmod-load.sh

## AMI NVIDIA EKS Bottlerocket

Quando si utilizza l'operatore GPU NVIDIA con le AMI NVIDIA EKS-optimized Bottlerocket, è necessario disabilitare l'installazione da parte dell'operatore del driver, del toolkit e del plug-in del dispositivo poiché questi sono già inclusi nelle AMI EKS.

Oltre ai componenti EKS AMI standard, le AMI NVIDIA Bottlerocket includono i seguenti componenti. EKS-optimized Le dipendenze minime per EFA (kernel module e rdma-core) sono installate in tutte le varianti di Bottlerocket.

- Plugin per dispositivi NVIDIA Kubernetes
- Driver NVIDIA
- Driver in modalità utente NVIDIA CUDA
- Toolkit per contenitori NVIDIA
- NVIDIA fabric manager
- NVIDIA ha persistito
- Driver NVIDIA IMEX
- NVIDIA NVLink Subnet Manager
- Gestore NVIDIA MIG

Per dettagli sul driver in modalità utente NVIDIA CUDA e sul CUDA runtime/libraries utilizzato nei contenitori delle applicazioni, consulta la documentazione di NVIDIA. https://docs.nvidia.com/deploy/cuda-compatibility/why-cuda-compatibility.html#why-cuda-compatibility La versione CUDA mostrata `nvidia-smi` è la versione del driver in modalità utente NVIDIA CUDA installato sull'host, che deve essere compatibile con il CUDA utilizzato nei contenitori di applicazioni. runtime/libraries

Consulta le informazioni sulla versione di Bottlerocket nella documentazione di Bottlerocket per dettagli sui pacchetti installati e sulle relative versioni. Le AMI NVIDIA EKS-optimized Bottlerocket supportano il kernel 6.12 per le versioni di Kubernetes 1.33 e successive e la versione del driver NVIDIA 580 per le versioni Kubernetes 1.34 e successive. Il driver NVIDIA 580 è necessario per utilizzare CUDA 13+.

## EKS-optimized AMI Neuron

Per dettagli su come eseguire carichi di lavoro di addestramento e inferenza utilizzando Neuron con Amazon EKS, consulta i seguenti riferimenti:

- Contenitori - Kubernetes - Guida introduttiva nella documentazione di Neuron AWS
- Esempio di formazione in AWS Neuron EKS Samples on GitHub
- Distribuisci carichi di lavoro di inferenza ML con Inferentia su Amazon EKS

Per trovare le AMI EKS-optimized Neuron più recenti, consulta e. the section called “Ottieni le ultime novità IDs” the section called “Ottenimento degli ID della versione più recente”

Quando usi Amazon Elastic Fabric Adapter (EFA) con le AMI EKS-optimized AL2023 o Bottlerocket Neuron, devi installare il plug-in del dispositivo EFA separatamente. Per ulteriori informazioni, consulta the section called "Configurazione dei cluster di formazione con EFA".

## AMI EKS AL2023 Neuron

Le AMI Neuron EKS-optimized AL2023 non includono il driver Neuron DRA, il plug-in del dispositivo Neuron Kubernetes o l'estensione dello scheduler Neuron Kubernetes e devono essere installate separatamente. https://awsdocs-neuron.readthedocs-hosted.com/en/latest/containers/tutorials/k8s-neuron-scheduler.html Per ulteriori informazioni, consulta the section called "Acceleratori Neuron".

Oltre ai componenti EKS AMI standard, le AMI Neuron AL2023 includono i seguenti componenti. EKS-optimized

- Driver Neuron (aws-neuronx-dkms)
- Strumenti Neuron (aws-neuronx-tools)
- EFA minimal (modulo kernel e rdma-core)

Consulta lo script di installazione EKS AL2023 Neuron AMI per dettagli su come le AMI EKS configurano le dipendenze Neuron. Consulta le versioni EKS-optimized AL2023 GitHub per vedere le versioni dei componenti incluse nelle AMI. Puoi trovare l'elenco dei pacchetti installati e le relative versioni su un'istanza EC2 in esecuzione con il comando. `dnf list installed`

## AMI EKS Bottlerocket Neuron

Le varianti standard di Bottlerocket (aws-k8s) includono le dipendenze Neuron che vengono rilevate e caricate automaticamente durante l'esecuzione su istanze Inferentia o Trainium EC2. AWS

Le AMI EKS-optimized Bottlerocket non includono il driver Neuron DRA, il plug-in del dispositivo Neuron Kubernetes o l'estensione dello scheduler Neuron Kubernetes e devono essere installate separatamente. https://awsdocs-neuron.readthedocs-hosted.com/en/latest/containers/tutorials/k8s-neuron-scheduler.html Per ulteriori informazioni, consulta the section called "Acceleratori Neuron".

Oltre ai componenti EKS AMI standard, le AMI Bottlerocket Neuron includono i seguenti componenti. EKS-optimized

- Driver Neuron (aws-neuronx-dkms)
- EFA minimal (modulo kernel e rdma-core)

Quando si utilizzano le AMI EKS-optimized Bottlerocket con istanze Neuron, è necessario configurare quanto segue nei dati utente di Bottlerocket. Questa impostazione consente al contenitore di assumere la proprietà del dispositivo Neuron montato in base ai valori e ai valori forniti nelle specifiche del carico di lavoro. `runAsUser` `runAsGroup` Per ulteriori informazioni sul supporto di Neuron in Bottlerocket, consulta il file Quickstart on EKS readme on. https://github.com/bottlerocket-os/bottlerocket/blob/develop/QUICKSTART-EKS.md#neuron-support GitHub

```
[settings]
[settings.kubernetes]
device-ownership-from-security-context = true
```

Consulta il changelog del kit del kernel Bottlerocket per informazioni sulla versione del driver Neuron inclusa nelle AMI Bottlerocket. EKS-optimized

## Esegui GPU-accelerated contenitori (Windows su EC2) G-Series

> **Important**
>
> Il Kubernetes Device Plugin for DirectX by TensorWorks è uno strumento di terze parti che non è approvato, supportato o gestito da. AWS AWS non si assume alcuna responsabilità per la sicurezza, l'affidabilità o le prestazioni di questo plugin.

Scopri come eseguire carichi di lavoro con container GPU-accelerated Windows su Amazon EKS (Elastic Kubernetes Service) utilizzando le GPU NVIDIA con il Kubernetes Device Plugin for DirectX by. TensorWorks Per ulteriori informazioni, consulta Kubernetes Device Plugin for DirectX.

Esistono due approcci principali alla configurazione dei contenitori Windows: GPU-acceleration

- Opzione 1: crea un'AMI EKS ottimizzata per Windows personalizzata con i driver GPU richiesti preinstallati.
  - Usa questo approccio quando hai bisogno di un ambiente coerente e preconfigurato pronto per eseguire contenitori GPU-accelerated Windows e sei in grado di investire gli sforzi aggiuntivi per creare e mantenere l'AMI personalizzata.
- Opzione 2: installa i driver GPU necessari sui nodi worker EKS dopo aver avviato l'istanza.
  - Utilizza questo approccio quando desideri un processo di configurazione più semplice e puoi installare i driver GPU su ogni nuovo nodo worker. Più adatto a un ambiente di sviluppo quando si valutano o si creano GPU-accelerated prototipi di carichi di lavoro.

Entrambi gli approcci possono essere sfruttati utilizzando i passaggi descritti in questa guida.

## Considerazioni

Questa guida fornisce i passaggi per installare e configurare i contenitori Windows utilizzando GPU-acceleration le GPU NVIDIA, i driver NVIDIA GRID e il Kubernetes Device Plugin for DirectX by. TensorWorks I passaggi sono stati testati e verificati per soddisfare i carichi di lavoro dei container Windows su GPU-acceleration Amazon EKS. Per ulteriori informazioni sui driver e plug-in dei dispositivi compatibili, consulta the section called "Limiti noti". Prima di procedere, tieni presente i seguenti punti:

- Solo i tipi di G-family istanze con driver NVIDIA GRID sono stati testati e verificati per funzionare con questa guida. Sebbene anche altri tipi di istanze e combinazioni di driver possano essere in grado di eseguire contenitori GPU-accelerated Windows, potrebbero richiedere passaggi di configurazione aggiuntivi non trattati in questa guida.
- Solo DirectX-based i carichi di lavoro sono stati testati e verificati per funzionare con questa guida. Sebbene altre API GPU come OpenGL, Vulkan e OpenCL possano essere potenzialmente compatibili per eseguire contenitori GPU-accelerated Windows, potrebbero richiedere passaggi di configurazione aggiuntivi non descritti in questa guida.
- Esistono alcune limitazioni note di cui tenere conto prima di eseguire i contenitori Windows. GPU-accelerated Consulta la sezione the section called "Limiti noti" per ulteriori informazioni.

## Prerequisiti

Per abilitare l'accelerazione GPU per i container Windows su Amazon EKS, devi preparare i seguenti requisiti prima di procedere:

- Avvia un cluster Amazon EKS con Kubernetes v1.27 o successive.
- Esegui il provisioning dei nodi Windows con Windows Server 2022 o versioni successive.
- Effettua il provisioning G-family dei nodi Windows nei tipi di istanza, ad esempio G4 o https://aws.amazon.com/ec2/instance-types/g5/ G5.
- Esegui il provisioning dei nodi Windows con un runtime dei container con containerd `1.7.x` o `2.x.x`. (Vedi the section called "Ottenere informazioni sulla versione" per verificare la versione del containerd nella tua AMI ottimizzata di Amazon EKS).

## Installa il driver GPU su ogni nodo Windows

Per installare i driver NVIDIA GRID sui nodi worker EKS, segui i passaggi descritti in NVIDIA drivers for your Amazon EC2 instance. Vai a Installation options - Option 3: GRID drivers e segui i passaggi per l'installazione.

### Installa per Windows Server Core

Per Windows Server Core, che non offre un'esperienza desktop, installa i driver NVIDIA GRID in modo invisibile utilizzando i seguenti comandi:

```
$nvidiaInstallerFilePath = nvidia-driver-installer.exe # Replace with path to installer
$installerArguments = "-s -clean -noreboot -noeula"
Start-Process -FilePath $nvidiaInstallerFilePath -ArgumentList $installerArguments -
Wait -NoNewWindow -PassThru
```

### Verifica dell'installazione

Esegui il PowerShell comando seguente per visualizzare le informazioni diagnostiche sulle GPU sull'istanza:

```
nvidia-smi
```

Questo comando mostra la versione del driver NVIDIA e le informazioni sull'hardware della GPU. Assicurati che l'output di questo comando corrisponda alla versione del driver NVIDIA GRID che prevedi di installare.

## Implementa il plug-in del dispositivo GPU su ogni nodo

Per consentire il rilevamento e l'esposizione delle risorse della GPU ai container sui nodi Windows, è necessario un plug-in per il dispositivo. Distribuisci il DirectX Device Plugin di Tensorworks su ogni nodo di lavoro eseguendolo come se fosse nel tuo cluster EKS. DaemonSet Segui la guida all'installazione specificata in README.md, che comporterà i seguenti passaggi. Si consiglia di:

- Implementare il plug-in del dispositivo nel namespace `kube-system`.
- Imposta limiti di risorse appropriati DaemonSet per assicurarti che non consumi risorse eccessive sui tuoi nodi.

> **Note**
>
> Il plug-in del dispositivo DaemonSet verrà eseguito su ogni nodo come contenitore di processi host con privilegi elevati. Si consiglia di implementare i controlli RBAC per limitarne l'accesso in modo che solo gli utenti autorizzati DaemonSet possano eseguire comandi privilegiati.

Quando si eseguono GPU-accelerated contenitori, il plug-in del dispositivo supporta due modalità:

- Single-tenancy modalità: questa modalità dedica tutte le risorse della GPU a un singolo contenitore sull'istanza. Installa i plug-in del dispositivo con supporto della tenancy singola utilizzando il comando seguente: README.md Per ulteriori informazioni, consulta.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/default-daemonsets.yml"
  ```

- Multi-tenancy modalità: questa modalità consente di condividere le risorse GPU tra più contenitori sull'istanza. Installa i plug-in del dispositivo con supporto della multi-tenancy utilizzando il comando seguente: README.md Per ulteriori informazioni, consulta.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-inline.yml"
  ```

  In alternativa, utilizzare a ConfigMap per specificare la multi-tenancy.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-configmap.yml"
  ```

Verifica dell'implementazione del plug-in del dispositivo

Dopo aver distribuito il plug-in del dispositivo, sostituisci `<namespace>` ed esegui il comando seguente per verificare che il plug-in del dispositivo DirectX funzioni correttamente su tutti i nodi Windows.

```
kubectl get ds device-plugin-wddm -n <namespace>
```

Verifica che i container siano pronti per l'implementazione

Una volta che il plug-in del dispositivo DaemonSet è in esecuzione sui nodi di lavoro di GPU-powered Windows, utilizzate il comando seguente per verificare che ogni nodo disponga di GPU allocabili. Il numero corrispondente deve corrispondere al numero di dispositivi DirectX su ciascun nodo.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,DirectX:.status.allocatable.directx\.microsoft\.com/
display"
```

## Esecuzione di contenitori Windows con GPU-acceleration

Prima di avviare i pod, specifica il nome della risorsa `directx.microsoft.com/display` in `.spec.containers[].resources`. Ciò indicherà che i contenitori richiedono GPU-enabled funzionalità e `kube-scheduler` tenterà di posizionare i pod nel nodo Windows preconfigurato con le risorse GPU disponibili.

Vedi il comando di esempio riportato di seguito che avvia `Job` per eseguire una simulazione Monte Carlo e stimare il valore di pi. Questo esempio proviene dal GitHub repository Kubernetes Device Plugins for DirectX, che contiene diversi esempi tra cui scegliere che puoi eseguire per testare le funzionalità della GPU del nodo Windows.

```
cat <<EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: example-cuda-montecarlo-wddm
spec:
  template:
    spec:
      containers:
      - name: example-cuda-montecarlo-wddm
        image: "index.docker.io/tensorworks/example-cuda-montecarlo:0.0.1"
        resources:
          limits:
            directx.microsoft.com/display: 1
      nodeSelector:
        "kubernetes.io/os": windows
      restartPolicy: Never
  backoffLimit: 0
EOF
```

## Limiti noti

Tutte le GPU sono utilizzabili

Tutte le GPU dell'istanza saranno utilizzabili da ogni container in esecuzione sull'host, anche quando si richiede un numero specifico di GPU per un determinato container. Inoltre, il comportamento predefinito prevede che tutti i container in esecuzione sull'host utilizzino la GPU con indice 0, anche se sul nodo sono disponibili più GPU. Pertanto, affinché le attività con più GPU funzionino correttamente, è necessario designare esplicitamente il dispositivo GPU specifico da utilizzare all'interno del codice dell'applicazione.

L'implementazione esatta per allocare un dispositivo da utilizzare per l'applicazione dipenderà dal linguaggio di programmazione o dal framework in uso. Ad esempio, se utilizzi la programmazione CUDA, per selezionare una GPU specifica, puoi specificare esplicitamente il dispositivo da utilizzare nel codice dell'applicazione utilizzando la funzione cuda (). SetDevice

La necessità di specificare in modo esplicito il dispositivo è dovuta a un problema noto che interessa i container Windows. Puoi tenere traccia dei progressi nella risoluzione di questo problema nel numero #333 di -Containers. microsoft/Windows La tabella seguente è una rappresentazione visiva e un esempio pratico di questo comportamento di allocazione delle GPU.

Consideriamo uno scenario in cui esiste un singolo nodo Windows del tipo di istanza EC2 `g4dn.12xlarge`, dotato di quattro GPU. Consideriamo uno scenario in cui vengono lanciati tre pod su questa istanza. La tabella mostra che, indipendentemente dal numero di GPU richieste da ciascun container, tutti e tre i pod hanno accesso a tutte e quattro le GPU dell'istanza e, per impostazione predefinita, utilizzeranno la GPU con indice di dispositivo 0.

| Pod | GPU richieste | Accesso effettivo alla GPU | Utilizzo predefinito della GPU | Indici disponibili della GPU | Istanze totali della GPU |
|---|---|---|---|---|---|
| Pod 1 | 1 GPU | Tutte e 4 le GPU | GPU con indice 0 | 0, 1, 2, 3 | 4 |
| Pod 2 | 2 GPU | Tutte e 4 le GPU | GPU con indice 0 | 0, 1, 2, 3 | 4 |
| Pod 3 | 1 GPU | Tutte e 4 le GPU | GPU con indice 0 | 0, 1, 2, 3 | 4 |

Supporto per il plug-in dei dispositivi Kubernetes

L'implementazione ufficiale di NVIDIA del Kubernetes device plugin non supporta Windows. Puoi tenere traccia dei progressi nell'aggiunta del supporto ufficiale per Windows nel NVIDIA/k8s-device-plugin numero #419.

Limitazioni delle istanze di calcolo GPU

A seconda della configurazione AWS dell'account, potresti avere limiti di servizio sul numero e sui tipi di istanze di calcolo GPU Amazon EC2 che puoi avviare. Se hai bisogno di capacità aggiuntiva, consulta Request a quota increase.

È necessario creare un'AMI ottimizzata per GPU Windows

Amazon EKS non fornisce alcuna AMI EKS ottimizzata per GPU Windows né un componente gestito da EC2 Image Builder. È necessario seguire i passaggi di questa guida per creare un'AMI EKS ottimizzata per Windows e personalizzata con i driver GPU richiesti preinstallati o installare i driver GPU necessari sui nodi worker EKS dopo l'avvio delle istanze.

Inferentia e Trainium non sono supportati

I carichi di lavoro basati su AWS Inferentia e AWS Trainium non sono supportati su Windows.

## Esecuzione dei corsi di machine learning su Amazon EKS con Elastic Fabric Adapter

In questo argomento viene descritto come integrare Elastic Fabric Adapter (EFA) con i pod implementati nel cluster Amazon EKS. Elastic Fabric Adapter (EFA) è un'interfaccia di rete per le istanze Amazon EC2 che consente di eseguire applicazioni che richiedono livelli elevati di comunicazioni internodale su larga scala in AWS. La sua interfaccia hardware di bypass del sistema operativo personalizzata migliora le prestazioni delle comunicazioni tra istanze, che è fondamentale per dimensionare queste applicazioni. Con EFA, le applicazioni High Performance Computing (HPC) che utilizzano le applicazioni MPI (Message Passing Interface) e Machine Learning (ML) e che utilizzano NVIDIA Collective Communications Library (NCCL) possono dimensionare fino a migliaia di CPU o GPU. Di conseguenza, ottieni le prestazioni delle applicazioni dei cluster HPC locali con l'elasticità e la flessibilità on-demand del cloud. AWS L'integrazione di EFA con le applicazioni in esecuzione su cluster Amazon EKS può ridurre il tempo necessario per completare carichi di lavoro di formazione distribuiti su larga scala senza dover aggiungere ulteriori istanze al cluster. Per ulteriori informazioni su EFA, consulta Elastic Fabric Adapter.

Per informazioni sull'installazione e la gestione dei dispositivi EFA nel cluster EKS, tra cui il driver EFA DRA, il plug-in del dispositivo EFA, l'allocazione basata sulla topologia e la configurazione dell'interfaccia di rete, consulta. the section called "Dispositivi EFA"

## Tipi di istanze con EFA

L' AWS EFA Kubernetes Device Plugin supporta tutti i tipi di istanze Amazon EC2 con EFA. Per visualizzare un elenco di tutti i tipi di istanze con EFA, consulta Tipi di istanze supportati nella Guida per l'utente di Amazon EC2. Tuttavia, per eseguire rapidamente le applicazioni ML, consigliamo che un'istanza disponga di chip di accelerazione hardware come GPU NVIDIA, chip AWS Inferentia o chip Trainium, oltre all'EFA.AWS Per visualizzare un elenco di tipi di istanze con chip di accelerazione hardware ed EFA, consulta Calcolo accelerato nella Guida per l'utente di Amazon EC2.

Nel confrontare i tipi di istanza e scegliere tra di essi, considera il numero di schede di rete EFA disponibili per quel tipo di istanza, nonché il numero di schede di accelerazione, la quantità di CPU e la quantità di memoria. È possibile assegnare fino a un EFA per scheda di rete. Un EFA conta come un'interfaccia di rete. Per vedere quante EFA sono disponibili per ogni tipo di istanza con EFA, consulta l'elenco delle schede di rete nella Guida per l'utente di Amazon EC2.

## EFA e interfacce EFA-only

Un Elastic Fabric Adapter (EFA) è un'interfaccia di rete che combina le funzionalità di un Elastic Network Adapter (ENA) e un' OS-bypass interfaccia, basata sul protocollo AWS Scalable Reliable Datagram (SRD). Le funzionalità EFA consentono alle applicazioni di comunicare direttamente con l'hardware per un trasporto a bassa latenza. È possibile scegliere di accedere solo alle funzionalità EFA utilizzando le EFA-onlyinterfacce, limitando la comunicazione alle interfacce all'interno della stessa zona di disponibilità.

Per creare nodi che possono avere EFA-only interfacce, è necessario utilizzare un EC2 Launch Template personalizzato e impostarlo su. `InterfaceType` `efa-only` Nel tuo Launch Template personalizzato, non puoi impostare la scheda `0` di rete come EFA-only interfaccia, in quanto si tratta della scheda di rete principale e dell'interfaccia di rete dell'istanza EC2. È necessario disporre della versione VPC CNI `1.18.5` o successiva per le interfacce. EFA-only Se utilizzi Amazon Linux 2, la versione ami deve essere `v20240928` o successiva per le EFA-only interfacce.

La seguente procedura ti guida alla creazione di un cluster EKS `eksctl` con nodi dotati di GPU NVIDIA e interfacce EFA. Non è possibile utilizzarlo `eksctl` per creare nodi e gruppi di nodi che utilizzano interfacce. EFA-only

## Prerequisiti

- Un cluster Amazon EKS esistente. Se non disponi di un cluster esistente, creane uno utilizzandoNozioni di base. Il cluster deve essere implementato in un VPC con almeno una sottorete privata con indirizzi IP disponibili sufficienti in cui implementare i nodi. La sottorete privata deve disporre di un accesso Internet in uscita fornito da un dispositivo esterno, ad esempio un gateway NAT.

  Se prevedi di utilizzare `eksctl` per creare il gruppo di nodi, `eksctl` può anche creare un cluster per tuo conto.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum`apt-get, o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called “Configura kubectl ed eksctl”:
- Prima di avviare i nodi worker che supportano più di un Elastic Fabric Adapter, come ad esempio `p4d` o `p5`, è necessario aver installato la versione `1.7.10` o successiva del plugin CNI di Amazon VPC per Kubernetes. Per ulteriori informazioni sull'aggiornamento della versione del plugin CNI di Amazon VPC per Kubernetes, consulta the section called “CNI di Amazon VPC”.
- Per le istanze p6-b200, è necessario utilizzare la versione del plugin del dispositivo EFA v0.5.6 o successiva.

> **Important**
>
> Un'importante considerazione necessaria per l'adozione di EFA con Kubernetes è la configurazione e la gestione di Huge Pages come risorsa nel cluster. Per ulteriori informazioni, consultare Gestione di Huge Pages nella documentazione Kubernetes. Le

istanze Amazon EC2 con il driver EFA installato assegnano in via anticipata 5.128 Huge Pages da 2 MiB, che possono essere richieste come risorse da utilizzare nelle specifiche del processo.

## Creazione di un gruppo di nodi

La procedura seguente consente di creare un gruppo di nodi con un gruppo di nodi supportato da `p4d.24xlarge` con interfacce EFA e GPUDirect RDMA, e di eseguire un test di esempio NVIDIA Collective Communications Library (NCCL) per prestazioni NCCL multi-nodo utilizzando più EFA. L'esempio può essere utilizzato come modello per la formazione distribuita di deep learning su Amazon EKS utilizzando EFA.

1. Determina quali tipi di istanze Amazon EC2 che supportano EFA sono disponibili nella AWS regione in cui desideri distribuire i nodi. Sostituiscilo *`region-code`* con la AWS regione in cui desideri distribuire il gruppo di nodi.

   ```
   aws ec2 describe-instance-types --region region-code \
       --filters Name=network-info.efa-supported,Values=true \
       --query "InstanceTypes[*].[InstanceType]" --output text
   ```

   Quando distribuisci i nodi, il tipo di istanza che desideri distribuire deve essere disponibile nella AWS regione in cui si trova il cluster.

2. Determina in quali zone di disponibilità è disponibile il tipo di istanza che desideri implementare. In questo tutorial, viene utilizzato il tipo di `p5.48xlarge` istanza che deve essere restituito nell'output per la AWS regione specificata nel passaggio precedente. Quando distribuisci i nodi in un cluster di produzione, sostituiscili *`p5.48xlarge`* con qualsiasi tipo di istanza restituito nel passaggio precedente.

   ```
   aws ec2 describe-instance-type-offerings --region region-code \
       --location-type availability-zone --filters Name=instance-
   type,Values=p4d.24xlarge,p5.48xlarge \
       --query 'InstanceTypeOfferings[*].Location' --output text
   ```

   Di seguito viene riportato un output di esempio.

   ```
   us-west-2a    us-west-2c    us-west-2b
   ```

Prendi nota delle zone di disponibilità restituite per l'uso nelle fasi successive. Quando implementi i nodi in un cluster, il tuo VPC deve disporre di sottoreti con indirizzi IP disponibili in una delle zone di disponibilità restituite nell'output.

3. Crea un gruppo di nodi utilizzando `eksctl`. È necessaria la versione `0.215.0` o una versione successiva dello strumento da riga di `eksctl` comando installata sul dispositivo o AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

   a. Copiare i contenuti seguenti in un file denominato *`efa-cluster.yaml`*. Sostituire i valori di esempio con i propri valori. Puoi sostituire `p5.48xlarge` con un'istanza diversa, ma in questo caso assicurati che i valori per `availabilityZones` corrispondano a zone di disponibilità restituite per il tipo di istanza nel passaggio 1.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

   b. Crea un gruppo di nodi gestito in un cluster esistente.

```
eksctl create nodegroup -f efa-cluster.yaml
```

Se non disponi di un cluster esistente, puoi eseguire il comando seguente per creare un cluster e il gruppo di nodi.

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> Poiché il tipo di istanza utilizzato in questo esempio ha delle GPU, `eksctl` installa automaticamente il plugin del dispositivo NVIDIA Kubernetes su ogni istanza per tuo conto durante l'utilizzo di Amazon Linux 2. Ciò non è necessario per Bottlerocket, poiché il plugin del dispositivo NVIDIA è integrato nella variante NVIDIA EKS di Bottlerocket. Quando `efaEnabled` è impostato su `true` nella configurazione del gruppo di nodi, `eksctl` distribuirà automaticamente anche il plugin del dispositivo EFA sui nodi.

Utilizzo di Bottlerocket con EFA

La versione AMI di Bottlerocket 1.28.0 e successive includono il supporto ufficiale per EFA. Per utilizzare Bottlerocket per EFA-enabled i nodi, specificalo `amiFamily: Bottlerocket` nella configurazione. Se è necessario utilizzare un ID AMI personalizzato, è necessario utilizzare `nodeGroups` standard anziché `managedNodeGroups`.

Ecco un esempio di configurazione:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-bottlerocket-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
```

```
  - name: my-efa-bottlerocket-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
    amiFamily: Bottlerocket
    bottlerocket:
      enableAdminContainer: true
      settings:
        kernel:
          sysctl:
            "vm.nr_hugepages": "3000"  # Configures 3000 * 2Mi = 6000Mi hugepages
```

L'impostazione sysctl `vm.nr_hugepages` riportata sopra configura il numero di Huge Pages di 2 Mi. In questo esempio, 3000 significa 3000 * 2 Mi = 6000 Mi di Huge Pages.

Verifica l'installazione del plugin del dispositivo EFA

Quando crei un gruppo di nodi con `efaEnabled: true`, `eksctl` distribuisce automaticamente il plugin per dispositivi EFA Kubernetes per te. Puoi verificare che il plugin del dispositivo sia installato e funzioni correttamente:

1. Controlla lo stato: DaemonSet

   ```
   kubectl get daemonsets -n kube-system
   ```

   Output di esempio:

   ```
   NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE
    AVAILABLE   NODE SELECTOR   AGE
   aws-efa-k8s-device-plugin-daemonset   2         2         2       2            2
         <none>          6m16s
   ...
   ```

   Qui, il plug-in del dispositivo EFA DaemonSet è in esecuzione su due nodi. Entrambi sono PRONTO e DISPONIBILE.
2. Successivamente, verifica i pod creati da: DaemonSet

```
kubectl get pods -n kube-system -l name=aws-efa-k8s-device-plugin
```

Output di esempio:

```
NAME                                          READY   STATUS    RESTARTS   AGE
aws-efa-k8s-device-plugin-daemonset-d68bs   1/1     Running   0          6m16s
aws-efa-k8s-device-plugin-daemonset-w4l8t   1/1     Running   0          6m16s
```

I pod dei plugin del dispositivo EFA sono nello stato In esecuzione, a conferma che il plugin è stato distribuito correttamente ed è operativo.

3. Verifica la registrazione delle risorse:

   Puoi confermare che la risorsa `vpc.amazonaws.com/efa` è registrata con il kubelet descrivendo i nodi:

   ```
   kubectl describe nodes
   ```

   se la risorsa EFA è registrata correttamente, la vedrai elencata nelle risorse di capacità e assegnabilità del nodo. Esempio:

   ```
   Capacity:
     ...
     vpc.amazonaws.com/efa:  4
   Allocatable:
     ...
     vpc.amazonaws.com/efa:  4
   ```

   questo output conferma che il nodo riconosce la risorsa EFA, rendendola disponibile per i pod che la richiedono.

## (Facoltativo) Test delle prestazioni dell'EFA

Consigliamo di testare la configurazione EFA. È possibile utilizzare i test NCCL presenti nel repository su. `aws-samples/awsome-distributed-training` GitHub I test NCCL valutano le prestazioni della rete utilizzando la libreria NVIDIA Collective Communications. La procedura seguente prevede l'invio dei test NCCL su Amazon EKS.

1. Implementa l'operatore MPI Kubeflow:

   Per i test NCCL è possibile applicare l'operatore Kubeflow MPI. L'operatore MPI semplifica l'esecuzione di corsi di formazione Allreduce-style distribuiti su Kubernetes. Per ulteriori informazioni, consulta MPI Operator on. GitHub

2. Eseguite il test delle prestazioni NCCL multinodo per verificare: GPUDirectRDMA/EFA

   Per verificare le prestazioni NCCL con GPUDirectRDMA su EFA, esegui il test delle prestazioni NCCL standard. Per ulteriori informazioni, consulta il repository ufficiale NCCL-Testssu. GitHub

   Completa i seguenti passaggi per eseguire un test delle prestazioni NCCL a due nodi. Nell'esempio del processo di test NCCL, ogni lavoratore richiede otto GPU, 5120 Mi di, quattro EFA e 8000 Mi di `hugepages-2Mi` memoria, il che significa effettivamente che ogni lavoratore consuma tutte le risorse di un'istanza. `p5.48xlarge`

   a. Crea il manifesto MPIJob:

      Copia quanto riportato di seguito in un file denominato `nccl-tests.yaml`.

```
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nccl-tests
spec:
  runPolicy:
    cleanPodPolicy: Running
    backoffLimit: 20
  slotsPerWorker: 8
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
          restartPolicy: OnFailure
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: test-nccl-launcher
            env:
             - name: PATH
               value: $PATH:/opt/amazon/efa/bin:/usr/bin
            command:
```

```
            - /opt/amazon/openmpi/bin/mpirun
            - --allow-run-as-root
            - --tag-output
            - -np
            - "16"
            - -N
            - "8"
            - --bind-to
            - none
            - -x
            - PATH
            - -x
            - LD_LIBRARY_PATH
            - -x
            - NCCL_DEBUG=INFO
            - -x
            - NCCL_BUFFSIZE=8388608
            - -x
            - NCCL_P2P_NET_CHUNKSIZE=524288
            - -x
            - NCCL_TUNER_PLUGIN=/opt/amazon/ofi-nccl/lib/x86_64-linux-gnu/libnccl-
ofi-tuner.so
            - --mca
            - pml
            - ^cm,ucx
            - --mca
            - btl
            - tcp,self
            - --mca
            - btl_tcp_if_exclude
            - lo,docker0,veth_def_agent
            - /opt/nccl-tests/build/all_reduce_perf
            - -b
            - "8"
            - -e
            - "16G"
            - -f
            - "2"
            - -g
            - "1"
            - -c
            - "1"
            - -n
            - "100"
```

```
    Worker:
      replicas: 2
      template:
        spec:
          nodeSelector:
            node.kubernetes.io/instance-type: "p5.48xlarge"
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: nccl-tests-worker
            volumeMounts:
            - name: shmem
              mountPath: /dev/shm
            resources:
              limits:
                nvidia.com/gpu: 8
                hugepages-2Mi: 5120Mi
                vpc.amazonaws.com/efa: 32
                memory: 32000Mi
              requests:
                nvidia.com/gpu: 8
                hugepages-2Mi: 5120Mi
                vpc.amazonaws.com/efa: 32
                memory: 32000Mi
          volumes:
          - name: shmem
            hostPath:
              path: /dev/shm
```

b. Applica il NCCL-tests MPIJob:

   Invia `MPIJob` applicando il manifesto. Ciò creerà due istanze Amazon EC2 `p5.48xlarge`.

```
kubectl apply -f nccl-tests.yaml
```

   Di seguito viene riportato un output di esempio.

```
mpijob.kubeflow.org/nccl-tests created
```

c. Verifica che il processo abbia avviato i pod:

   Visualizza i pod in esecuzione.

```
kubectl get pods
```

Di seguito viene riportato un output di esempio.

```
NAME                             READY   STATUS     RESTARTS   AGE
nccl-tests-launcher-nbql9    0/1     Init:0/1   0          2m49s
nccl-tests-worker-0          1/1     Running    0          2m49s
nccl-tests-worker-1          1/1     Running    0          2m49s
```

L'operatore MPI crea un pod di avvio e 2 pod worker (uno su ciascun nodo).

d. Verifica che il processo funzioni correttamente con i log:

Visualizza il log per il pod `nccl-tests-launcher`. Sostituisci *`nbql9`* con il valore dell'output.

```
kubectl logs -f nccl-tests-launcher-nbql9
```

Se il test è stato completato con successo, puoi distribuire le tue applicazioni che utilizzano la NVIDIA Collective Communications Library.

## Utilizzo AWS Istanze Inferentia con Amazon EKS per l'apprendimento automatico

In questo argomento viene descritto come creare un cluster Amazon EKS con nodi che eseguono istanze Amazon EC2 Inf1 e (facoltativo) come implementare un'applicazione di esempio. Le istanze Amazon EC2 Inf1 sono alimentate da chip AWS Inferentia, realizzati su misura per fornire prestazioni elevate e costi di inferenza più bassi nel cloud. AWS I modelli di apprendimento automatico vengono distribuiti nei contenitori utilizzando AWS Neuron, un kit di sviluppo software specializzato (SDK) composto da compilatore, runtime e strumenti di profilazione che ottimizzano le prestazioni di inferenza di apprendimento automatico dei chip Inferentia. AWS Neuron supporta i più diffusi framework di machine learning come,, e MXNet. TensorFlow PyTorch

> **Note**
>
> Gli ID logici dei dispositivi neuronali devono essere contigui. Se un pod che richiede più dispositivi Neuron è pianificato su un tipo di istanza `inf1.6xlarge` o `inf1.24xlarge` (che ha più di un dispositivo Neuron), il pod non si avvia se lo scheduler Kubernetes seleziona

ID dispositivo non contigui. Per ulteriori informazioni, consulta Device logical ID must be contiguous on. GitHub

## Prerequisiti

- Avere `eksctl` installato sul computer. Se non è installato, consulta Installation nella documentazione di `eksctl`.
- Avere `kubectl` installato sul computer. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".
- (Facoltativo) Avere `python3` installato sul computer. Se non lo hai installato, consultare Python downloads per le istruzioni di installazione.

## Creazione di un cluster

1. Creare un cluster con nodi di istanze Amazon EC2 Inf1. Puoi sostituire *`inf1.2xlarge`* con qualsiasi tipo di istanza Inf1. L'utility `eksctl` rileva l'avvio di un gruppo di nodi con un tipo di istanza `Inf1` e avvia i nodi usando una delle AMI Amazon Linux accelerate ottimizzate per Amazon EKS.

   > **Note**
   >
   > Non puoi usare i ruoli IAM per gli account di servizio con TensorFlow Serving.

   ```
   eksctl create cluster \
       --name inferentia \
       --region region-code \
       --nodegroup-name ng-inf1 \
       --node-type inf1.2xlarge \
       --nodes 2 \
       --nodes-min 1 \
       --nodes-max 4 \
       --ssh-access \
       --ssh-public-key your-key \
       --with-oidc
   ```

> **Note**
>
> Notare il valore della seguente riga dell'output. Viene utilizzato in un passaggio successivo (facoltativo).

```
[9]  adding identity "arn:aws: iam::111122223333:role/eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth ConfigMap
```

Quando si avvia un gruppo di nodi con `Inf1` istanze, installa `eksctl` automaticamente il plug-in del dispositivo AWS Neuron Kubernetes. Questo plugin pubblicizza i dispositivi Neuron come risorsa di sistema allo scheduler Kubernetes, che può essere richiesto da un container. Oltre alle policy IAM dei nodi Amazon EKS di default, viene aggiunta la policy di accesso di sola lettura Amazon S3 in modo che l'applicazione di esempio, che verrà trattata in un passaggio successivo, possa caricare un modello addestrato da Amazon S3.

2. Assicurarsi che tutti i pod siano stati avviati correttamente.

```
kubectl get pods -n kube-system
```

Output abbreviato:

```
NAME                                   READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

## (Facoltativo) Distribuisci un'immagine dell'applicazione Serving TensorFlow

Un modello addestrato deve essere compilato in un target Inferentia prima di poter essere implementato nelle istanze Inferentia. Per continuare, avrai bisogno di un TensorFlow modello ottimizzato per Neuron salvato in Amazon S3. Se non ne hai già uno SavedModel, segui il tutorial per creare un modello ResNet 50 compatibile con Neuron e carica il risultato SavedModel su S3. ResNet-50 è un popolare modello di apprendimento automatico utilizzato per le attività di riconoscimento delle immagini. Per ulteriori informazioni sulla compilazione dei modelli Neuron, consulta The AWS Inferentia Chip With DLAMI nella Deep Learning AMiS Developer Guide. AWS

Il manifesto di distribuzione di esempio gestisce un contenitore di servizio di inferenza predefinito fornito da Deep Learning Containers. TensorFlow AWS All'interno del contenitore si trovano il AWS Neuron Runtime e l'applicazione Serving. TensorFlow Un elenco completo dei contenitori di Deep Learning predefiniti ottimizzati per Neuron è disponibile nella GitHub sezione Immagini disponibili. All'avvio, il DLC recupererà il modello da Amazon S3, avvierà Neuron TensorFlow Serving con il modello salvato e attenderà le richieste di previsione.

Il numero di dispositivi Neuron assegnati all'applicazione di servizio può essere regolato modificando la risorsa `aws.amazon.com/neuron` nella implementazione yaml. Tieni presente che la comunicazione tra TensorFlow Serving e il runtime di Neuron avviene tramite GRPC, il che richiede il trasferimento della funzionalità al contenitore. `IPC_LOCK`

1. Aggiungere la policy `AmazonS3ReadOnlyAccess` IAM al ruolo di istanza del nodo creato nel passaggio 1 di Crea un cluster. Ciò è necessario affinché l'applicazione di esempio possa caricare un modello formato da Amazon S3.

   ```
   aws iam attach-role-policy \
       --policy-arn arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess \
       --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
   ```

2. Crea un file denominato `rn50_deployment.yaml` con i seguenti contenuti. Aggiornare il codice di Regione e il percorso del modello in modo che corrispondano alle impostazioni desiderate. Il nome del modello serve a scopo identificativo quando un client effettua una richiesta al TensorFlow server. Questo esempio utilizza il nome di un modello per corrispondere a uno script di esempio di ResNet 50 client che verrà utilizzato in una fase successiva per l'invio di richieste di previsione.

   ```
   aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --region
    us-west-2
   ```

   ```
   kind: Deployment
   apiVersion: apps/v1
   metadata:
     name: eks-neuron-test
     labels:
       app: eks-neuron-test
       role: master
   spec:
     replicas: 2
     selector:
   ```

```
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://${your-bucket-of-models}/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
```

```
            capabilities:
              add:
                - IPC_LOCK
```

3. Implementare il modello.

```
kubectl apply -f rn50_deployment.yaml
```

4. Crea un file denominato `rn50_service.yaml` con i seguenti contenuti. Le porte HTTP e gRPC vengono aperte per accettare le richieste di previsione.

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. Crea un servizio Kubernetes per la tua TensorFlow applicazione Model Serving.

```
kubectl apply -f rn50_service.yaml
```

## (Facoltativo) Fai previsioni sul tuo servizio Serving TensorFlow

1. Per testare localmente, inoltrare la porta gRPC al servizio `eks-neuron-test`.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. Creare uno script Python chiamato `tensorflow-model-server-infer.py` con il seguente contenuto. Questo script esegue l'inferenza tramite gRPC, che è un framework di servizi.

```
import numpy as np
import grpc
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc
from tensorflow.keras.applications.resnet50 import decode_predictions

if __name__ == '__main__':
   channel = grpc.insecure_channel('localhost:8500')
   stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
   img_file = tf.keras.utils.get_file(
       "./kitten_small.jpg",
       "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
   img = image.load_img(img_file, target_size=(224, 224))
   img_array = preprocess_input(image.img_to_array(img)[None, ...])
   request = predict_pb2.PredictRequest()
   request.model_spec.name = 'resnet50_neuron'
   request.inputs['input'].CopyFrom(
       tf.make_tensor_proto(img_array, shape=img_array.shape))
   result = stub.Predict(request)
   prediction = tf.make_ndarray(result.outputs['output'])
   print(decode_predictions(prediction))
```

3. Eseguire lo script per inviare previsioni al servizio.

```
python3 tensorflow-model-server-infer.py
```

Di seguito viene riportato un output di esempio:

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

# Gestisci l'elaborazione accelerata per i AI/ML carichi di lavoro su Amazon EKS

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS. AI/ML

Questa sezione spiega come acquistare e fornire istanze di calcolo accelerate EC2 per carichi di lavoro di AI/ML formazione e inferenza con Amazon EKS. Che tu stia addestrando modelli su larga scala, eseguendo inferenze in tempo reale o implementando applicazioni di intelligenza artificiale generativa, l'utilizzo della giusta GPU NVIDIA o della capacità di Trainium è fondamentale per le prestazioni dei tuoi carichi di lavoro. AWS

## Scegli tra i tipi di istanze EC2

Consulta le specifiche per le istanze di elaborazione accelerata di Amazon EC2 per i dettagli sulle istanze di calcolo accelerate di Amazon EC2 disponibili. Queste includono le istanze GPU NVIDIA prodotte da e gli acceleratori progettati da Trainium P-family e G-family Inferentia. AWS

## Scopri le opzioni di acquisto di EC2

Una volta che conosci le istanze accelerate di cui hai bisogno per i tuoi carichi di lavoro, il passaggio successivo è comprendere le opzioni di acquisto disponibili per l'acquisizione di questi tipi di istanze accelerate. AWS offre quattro opzioni di acquisto di capacità di elaborazione: On-Demand Instances, Spot Instances, Capacity Blocks for ML e Capacity Reservations (ODCR). On-Demand Ogni opzione soddisfa diversi modelli di carico di lavoro, profili di costo e requisiti di disponibilità. La documentazione sulle opzioni di acquisto di istanze Amazon EC2 spiega come funziona ciascuna opzione, il relativo modello di prezzo e quando utilizzarla.

- On-Demand Istanze: pagamento al secondo senza impegno e disponibilità immediata quando la capacità è disponibile. Ideale per lo sviluppo, la prototipazione, la scalabilità imprevedibile delle inferenze e qualsiasi carico di lavoro che richieda un calcolo immediato senza rischi di interruzione.
- Istanze Spot: fino al 90% di risparmio rispetto all'utilizzo di capacità EC2 inutilizzata, con un preavviso di interruzione On-Demand di 2 minuti. Ideale per carichi di lavoro tolleranti ai guasti che richiedono uno storage duraturo: ottimizzazione degli iperparametri, formazione distribuita con checkpoint periodici, inferenza batch e offline e pipeline di preelaborazione dei dati.

- Capacity Blocks for ML: istanze Reserve P-family e Trainium per una finestra fissa (24 ore, fino a 6 mesi), prenotate fino a 8 settimane di anticipo. Utilizza i Capacity Blocks per sessioni di formazione pianificate su larga scala, esperimenti di ottimizzazione limitati nel tempo e progetti di ricerca con scadenze note che richiedono un accesso prevedibile a un cluster di GPU.
- On-Demand Prenotazioni di capacità (ODCR): riserva capacità accelerata in una specifica zona di disponibilità senza un impegno a lungo termine, fatturata a tariffe standard indipendentemente dal fatto che la capacità venga utilizzata o meno. On-Demand Ideale per inferenze di produzione, SLA-bound servizi e applicazioni aziendali critiche in cui ritardi nella pianificazione o indisponibilità della capacità sono inaccettabili. A differenza dei Capacity Blocks, gli ODCR supportano entrambe le istanze. P-family G-family

## Abbina le opzioni di acquisto ai requisiti del carico di lavoro

Ora che hai compreso i tipi di istanze accelerate e le opzioni di acquisto, il passaggio successivo consiste nell'abbinare l'opzione di acquisto giusta ai requisiti specifici del carico di lavoro. I carichi di lavoro con maggiore flessibilità tra tipi di istanze, regioni e tempistiche consentono di ottenere più opzioni di acquisto e prezzi più bassi.

Basa la tua decisione su fattori come:

- Importanza strategica e impegni in materia di SLA
- Prevedibilità della domanda e flessibilità di pianificazione
- Disponibilità a impegnarsi in anticipo per una capacità riservata
- Flessibilità tra tipi di istanze, regioni e tempistiche
- Tolleranza alle interruzioni rispetto al risparmio sui costi

In pratica, i team adottano un approccio ibrido che combina diverse opzioni di acquisto per bilanciare costi, disponibilità e affidabilità in tutto il portafoglio di carichi di lavoro. L'articolo How to Get GPU Capacity on AWS fornisce un albero decisionale, confronti dei prezzi ed esempi reali per selezionare l'opzione di acquisto giusta per diversi tipi di carichi di lavoro.

## Verifica le quote dei servizi EC2

Prima di implementare qualsiasi opzione di acquisto di capacità sul tuo cluster EKS, verifica che il tuo AWS account disponga di una quota di vCPU sufficiente per le famiglie di istanze GPU che intendi utilizzare. Senza quote adeguate, Karpenter NodePools, EKS Auto Mode provisioning e i gruppi di

nodi EKS non riusciranno ad avviare nodi di elaborazione accelerati indipendentemente dall'opzione di acquisto selezionata.

AWS impone quote di vCPU separate per famiglia di istanze e modello di acquisto. Esamina le quote dei tipi di istanza Amazon EC2 per comprendere le quote predefinite per le istanze di calcolo accelerate.

Queste quote si basano sul numero di vCPU, non sul numero di istanze. Ad esempio, l'avvio di 10 istanze p6-b300.48xlarge richiede 1.920 vCPU (10 × 192). Le quote GPU predefinite sono spesso impostate a 0 per i nuovi account, quindi richiedi aumenti prima di tentare di distribuire le istanze.

Se riscontri limiti di quota durante la creazione di prenotazioni Capacity Block, l'avvio di On-Demand istanze o l'invio di richieste Spot, contatta l' AWS assistenza o il team del tuo AWS account per discutere dei tuoi requisiti ed esplorare le opzioni per garantire la capacità di elaborazione accelerata più adatta alle tue esigenze.

## Usa le opzioni di acquisto EC2 con Amazon EKS

Dopo aver selezionato un'opzione di acquisto di elaborazione accelerata EC2, configura il cluster Amazon EKS per utilizzare la capacità. Amazon EKS offre tre metodi di provisioning, ciascuno con un diverso equilibrio tra controllo e automazione:

- Amazon EKS Auto Mode: elaborazione AWS gestita che esegue automaticamente il provisioning, la scalabilità e l'applicazione di patch ai nodi. Utilizza Karpenter integrato per il provisioning e il sistema operativo Bottlerocket con driver NVIDIA e plug-in di dispositivo inclusi. Ideale quando si desidera un'infrastruttura gestita con un sovraccarico operativo minimo. Supporta il provisioning della capacità sia statico che dinamico.
- Karpenter (autogestito): progetto upstream open source che installi e gestisci nel tuo cluster Amazon EKS. Fornisce lo stesso modello di provisioning di EKS Auto Mode e hai il pieno controllo del sistema operativo, delle AMI, dell'ottimizzazione del kernel e del ciclo di vita dei nodi. Ideale per i team di piattaforma con requisiti che EKS Auto Mode non fornisce immediatamente.
- Gruppi di nodi (gestiti e autogestiti): supportata da EC2 Auto Scaling Groups (ASG), la capacità viene definita in anticipo tramite un modello di avvio EC2. Ideale per i team di piattaforma con gruppi di nodi esistenti gestiti o autogestiti da EKS e per i carichi di lavoro di formazione con dimensioni prevedibili con un footprint di calcolo statico e accelerato noto.

Le pagine seguenti descrivono in dettaglio ciascuna opzione di provisioning.

- the section called "EKS Auto Mode e Karpenter"
- the section called "Gruppi di nodi gestiti e nodi autogestiti"

## Strategia mista: combina le opzioni di acquisto

È comune combinare più opzioni di acquisto di capacità all'interno di un singolo cluster Amazon EKS. Questo approccio ottimizza contemporaneamente costi, disponibilità e affidabilità indirizzando diversi carichi di lavoro alla fonte di capacità più appropriata. I clienti implementano questa strategia ibrida utilizzando uno dei tre approcci di gestione del calcolo EKS (EKS Auto Mode, Karpenter o Node Groups) o li combinano all'interno dello stesso cluster.

EKS Auto Mode e Karpenter forniscono sempre prima la capacità riservata (ODCR e Capacity Block), seguita da Spot o. On-Demand Puoi abbinare questa priorità di provisioning delle istanze pianificando i carichi di lavoro critici sulla capacità riservata e i carichi di lavoro flessibili su Spot o sulle istanze. On-Demand Puoi controllare il routing dei carichi di lavoro tramite primitive di Kubernetes-native pianificazione: `nodeSelector` obiettivi per un tipo di capacità specifico, limiti e tolleranze, isolano le GPU NVIDIA o i nodi Trainium e distribuisci i carichi di lavoro nelle zone di disponibilità per un'elevata disponibilità. AWS `topologySpreadConstraints`

Un cluster Amazon EKS ben progettato organizza l'elaborazione accelerata NodePools o i gruppi di nodi in due categorie, Reserved e Burst, ciascuna allineata ai modelli di carico di lavoro più adatti alla strategia di capacità. Di seguito viene descritto un esempio.

- Capacità riservata: un gruppo `gpu-reserved` NodePool o un gruppo di nodi esegue l'inferenza di produzione e la formazione pianificata su larga scala sulla capacità riservata (ODCR e Capacity Block) per SLA-bound servizi e processi pianificati ad alta intensità di calcolo. Questo NodePool o il gruppo di nodi serve i carichi di lavoro di inferenza e produzione: endpoint di inferenza in tempo reale, gestione dei modelli di produzione e applicazioni aziendali critiche che richiedono una disponibilità di GPU sempre attiva con prestazioni prevedibili. Supporta anche lavori programmati ad alta intensità di calcolo: formazione distribuita pianificata, esperimenti di messa a punto su larga scala, progetti di ricerca limitati nel tempo e qualsiasi carico di lavoro di cui si conoscano in anticipo l'ora di inizio e la durata.
- Burst Capacity: un gruppo di nodi gestisce la sperimentazione, i `gpu-burst` NodePool carichi di lavoro ad hoc e l'elaborazione in batch. Utilizza le istanze Spot come tipo di capacità principale con riserva. On-Demand Questa combinazione massimizza i risparmi sui costi per i carichi di lavoro tolleranti ai guasti e garantisce la capacità quando Spot non è disponibile. Questa NodePool o il gruppo di nodi consente l'inferenza offline in batch, le pipeline di preelaborazione

dei dati, i processi di valutazione dei modelli, lo sviluppo e la prototipazione, il ridimensionamento imprevedibile delle inferenze, le sessioni di debug di breve durata e qualsiasi carico di lavoro che implementa il checkpoint e sia in grado di gestire le interruzioni di Spot o che non giustifichi una prenotazione ma non possa attendere le finestre riservate. I carichi di lavoro su questo NodePool o su un gruppo di nodi implementano il checkpoint e lo spegnimento graduale per gestire la perdita dei nodi entro la finestra di interruzione Spot di 2 minuti.

Specifica il tipo di capacità desiderato per i carichi di lavoro utilizzando NodeSelector:. `karpenter.sh/capacity-type: [spot, on-demand, reserved]` Weight-based il provisioning scala il cluster in modo efficiente su tutti i pool di capacità. Con questa architettura, puoi eseguire diversi AI/ML carichi di lavoro, dai notebook sperimentali all'inferenza di produzione, all'interno di un singolo cluster Amazon EKS ottimizzando i costi.

## Gestisci l'elaborazione per i AI/ML carichi di lavoro con EKS Auto Mode e Karpenter

> **Tip**
>
> AI/ML Registrati per i prossimi workshop Amazon EKS.

Questa sezione spiega come gestire l'elaborazione accelerata (AWS Trainium, GPU NVIDIA) per l'addestramento e i carichi di lavoro di inferenza AI utilizzando Amazon EKS Auto Mode o Karpenter autogestito.

EKS Auto Mode e Karpenter supportano due modalità di provisioning: provisioning dinamico e provisioning statico. Con il provisioning dinamico, EKS Auto Mode e Karpenter eseguono il provisioning e la scalabilità delle istanze di calcolo accelerate man mano che i carichi di lavoro vengono pianificati sul cluster. Con il provisioning statico, EKS Auto Mode e Karpenter forniscono e mantengono un numero fisso di nodi. Il provisioning dinamico e statico può essere utilizzato nello stesso cluster per mantenere un pool di capacità di base costante e scalare in base alle esigenze del carico di lavoro.

EKS Auto Mode e Karpenter supportano tutte e quattro le opzioni di acquisto della capacità (SpotOn-Demand, Capacity Blocks e ODCR) e forniscono sempre prima la capacità riservata, seguita da Spot o. On-Demand

## EKS Auto Mode e Karpenter

Entrambi gli approcci condividono l' NodePool API, ma si differenziano per proprietà operativa, API delle risorse, supporto del sistema operativo, gestione delle interruzioni Spot e flessibilità di configurazione.

| Funzionalità | Modalità automatica di EKS | Self-managed Karpenter |
|---|---|---|
| Ideale per | Team che preferiscono un'infras truttura gestita con un sovraccarico operativo minimo | Team che preferiscono il pieno controllo del ciclo di vita dei nodi, delle AMI, dell'ottimizzazione del sistema operativo e dell'applicazione delle patch. |
| Modello operativo | AWS fornisce e gestisce il controlle r Karpenter, GPU/Trainium i driver, i plug-in dei dispositivi, le patch del sistema operativo e la gestione delle interruzioni Spot. | L'utente installa e utilizza il controlle r Karpenter nel cluster e dispone di GPU/Trainium driver, plug-in di dispositivo, ciclo di vita dell'AMI, applicazione di patch e gestione delle interruzioni Spot. |
| Opzioni di calcolo | On-Demand, Spot, ODCRS, Capacity Blocks for ML | On-Demand, Spot, ODCRs, blocchi di capacità per ML |
| API per le risorse | `NodePool` (`karpenter.sh/v1`), `NodeClass` (`eks.amazonaws.com/v1`). | `NodePool` (`karpenter.sh/v1`), `EC2NodeClass` (`karpenter.k8s.aws/v1`). |
| Sistema operativo Node | Solo Bottlerocket. Dipendenze NVIDIA GPU, AWS Trainium ed EFA incluse. | AL2023, Bottlerocket, Windows o la tua AMI. |
| Durata del nodo | Durata massima del nodo di 21 giorni per l'applicazione delle patch di sicurezza. I carichi di lavoro devono tollerare la rotazione dei nodi. | Sei tu a definire il ciclo di vita dei nodi e i budget per le NodePool `expireAfter` interruzioni. |

| Funzionalità | Modalità automatica di EKS | Self-managed Karpenter |
|---|---|---|
| Gestione istantanea delle interruzioni | Nativo. Non è richiesta alcuna coda SQS o Node Termination Handler. | La tua responsabilità di configurare e abilitare. |
| Estrazione rapida dei contenitori | Pull parallelo SOCI incluso in tutte le istanze delle famiglie G, P e Trn | La tua responsabilità di configurare e abilitare. |
| Gruppi di collocamento EC2 | Cluster, partizione, diffusione | Cluster, partizione, diffusione |
| Configurazione dell'interfaccia di rete | Per configurazione dell'interfaccia per tipo `interface` o `EFA-only` | Per configurazione di interfaccia per tipo `interface` o `EFA-only` |
| Riparazione dei nodi | Abilitato per impostazione predefinita, è incluso l'agente di monitoraggio dei nodi EKS | Opzionalmente abilitato, agente di monitoraggio dei nodi EKS autogesti to |
| Prezzi | Tariffa di gestione EKS Auto Mode in aggiunta al costo dell'istanza EC2 sottostante. | Open source. Paghi per le istanze EC2 sottostanti. |

## Etichette note comuni AI/ML

EKS Auto Mode e Karpenter espongono le etichette delle istanze che è possibile utilizzare in Pod `nodeSelector` o `nodeAffinity` per indirizzare NodePool `requirements` i carichi di lavoro senza l'hardcoding dei tipi di istanza. Il prefisso dell'etichetta è diverso tra i due: EKS Auto Mode utilizza mentre Karpenter autogestito utilizza. `eks.amazonaws.com/` `karpenter.k8s.aws/`

Le tabelle seguenti mostrano le etichette pertinenti che possono essere utilizzate in. NodePools EKS Auto Mode e Karpenter applicano inoltre le etichette elencate nella documentazione di Karpenter ai nodi come parte del processo di provisioning che può essere ulteriormente utilizzato per il targeting dei carichi di lavoro.

EKS Auto Mode

Per l'elenco completo, vedi Etichette supportate da EKS Auto Mode. https://docs.aws.amazon.com/eks/latest/userguide/create-node-pool.html#auto-supported-labels

| Etichetta | Valore di esempio | Description |
|---|---|---|
| `eks.amazonaws.com/instance-family` | `p5` | Tipi di istanze con proprietà simili ma quantità di risorse diverse. |
| `eks.amazonaws.com/instance-category` | `p` | Categoria di istanza, in genere la lettera che precede il numero di generazione. |
| `eks.amazonaws.com/instance-generation` | `5` | Numero di generazione del tipo di istanza all'interno di una categoria. |
| `eks.amazonaws.com/instance-gpu-name` | `h100` | Nome della GPU sull'istanza. |
| `eks.amazonaws.com/instance-gpu-manufacturer` | `nvidia` | Nome del produttore della GPU. |
| `eks.amazonaws.com/instance-gpu-count` | `8` | Numero di GPU sull'istanza. |
| `eks.amazonaws.com/instance-gpu-memory` | `81920` | Mebibyte di memoria per GPU. |
| `karpenter.sh/capacity-type` | `reserved` | Tipo di capacità:`spot`,, o. `on-demand` `reserved` |
| `topology.kubernetes.io/zone` | `us-east-1a` | Zona di disponibilità. |

Self-managed Karpenter

Per l'elenco completo, consulta Karpenter Labels Well-Known .

| Etichetta | Valore di esempio | Description |
|---|---|---|
| `karpenter.k8s.aws/instance-family` | `p5` | Tipi di istanza con proprietà simili ma quantità di risorse diverse. |
| `karpenter.k8s.aws/instance-category` | `p` | Categoria di istanza, in genere la lettera che precede il numero di generazione. |
| `karpenter.k8s.aws/instance-generation` | `5` | Numero di generazione del tipo di istanza all'interno di una categoria. |
| `karpenter.k8s.aws/instance-gpu-name` | `h100` | Nome della GPU sull'istanza. |
| `karpenter.k8s.aws/instance-gpu-manufacturer` | `nvidia` | Nome del produttore della GPU. |
| `karpenter.k8s.aws/instance-gpu-count` | `8` | Numero di GPU sull'istanza. |
| `karpenter.sh/capacity-type` | `reserved` | Tipo di capacità:`spot`,`on-demand` , `oreserved`. |
| `topology.kubernetes.io/zone` | `us-east-1a` | Zona di disponibilità. |
| `kubernetes.io/arch` | `amd64` | Architettura della CPU. |

## Etichette di pianificazione per la capacità riservata

Quando EKS Auto Mode o Karpenter avviano un nodo in una prenotazione, aggiungono le seguenti etichette. Usali in`nodeSelector`, affinità dei nodi o NodePool requisiti per indirizzare i carichi di lavoro.

- `karpenter.sh/capacity-type`:`reserved`,`on-demand`, o. `spot` Indica la capacità di supporto del nodo.
- `karpenter.k8s.aws/capacity-reservation-id`: l'ID di prenotazione specifico in cui è stato avviato il nodo.
- `karpenter.k8s.aws/capacity-reservation-type`: `default` per gli ODCR, `capacity-block` per i blocchi di capacità.

Gli esempi seguenti mostrano i modelli di pianificazione più comuni:

Aggiungi un Pod a una prenotazione specifica (senza riserva):

```
spec:
  nodeSelector:
    karpenter.sh/capacity-type: reserved
    karpenter.k8s.aws/capacity-reservation-id: "cr-0123456789abcdef0"
```

Solo nodi ODCR di destinazione (qualsiasi ODCR, non blocchi di capacità):

```
spec:
  nodeSelector:
    karpenter.sh/capacity-type: reserved
    karpenter.k8s.aws/capacity-reservation-type: default
```

Scegli come target qualsiasi capacità riservata (ODCR o Capacity Block):

```
spec:
  nodeSelector:
    karpenter.sh/capacity-type: reserved
```

Preferisci riservato ma torna a Spot o, On-Demand se non disponibile:

```
spec:
```

```
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 100
          preference:
            matchExpressions:
              - key: karpenter.sh/capacity-type
                operator: In
                values: ["reserved"]
```

## Comportamento alla scadenza della prenotazione

Gli ODCR e i Capacity Block si comportano diversamente al termine della prenotazione. Assicurati che la tua strategia di pianificazione e checkpoint corrisponda al tipo di prenotazione che sostiene il tuo carico di lavoro.

### ODCR

Un'istanza avviata in un ODCR non rimane in tale ODCR a tempo indeterminato. L'ODCR può scadere, essere annullato oppure l'istanza può essere rimossa manualmente dall'ODCR. Se si verifica una di queste situazioni e EKS Auto Mode/Karpenter rileva che l'istanza non appartiene più a un ODCR, aggiorna l'etichetta del nodo da a. `karpenter.sh/capacity-type` `reserved` `on-demand` L'istanza continua a funzionare con On-Demand capacità standard e i Pod esistenti continuano a funzionare senza interruzioni.

> **Note**
>
> Qualsiasi Pod programmato con una restrizione non `nodeSelector: karpenter.sh/capacity-type: reserved` verrà programmato sul nodo se è stato rietichettato. Affinché i carichi di lavoro sopravvivano alla scadenza o all'annullamento dell'ODCR, utilizza lo `preferredDuringSchedulingIgnoredDuringExecution` schema mostrato sopra anziché un. `nodeSelector`

### Blocchi di capacità

A differenza degli ODCR, i Capacity Block hanno sempre un'ora di fine e EC2 termina le istanze Capacity Block 30 minuti prima dell'ora di fine (60 minuti per i tipi di istanza). UltraServer Pianifica i lavori di formazione e inferenza per completare o salvare lo stato prima della chiusura della finestra di prenotazione. I pod che utilizzano una restrizione `nodeSelector` per un determinato periodo

vengono eliminati una `capacity-reservation-id` `Pending` volta scaduto il blocco e non verranno riprogrammati altrove. Combina il checkpoint con il modello di affinità flessibile sopra riportato se hai bisogno che i carichi di lavoro passino ad altra capacità durante la scadenza del Capacity Block.

- Puoi utilizzare le istanze riservate fino a 30 minuti prima dell'ora di fine del Capacity Block per la maggior parte dei tipi di istanze o 60 minuti prima dell'ora di fine per i tipi di UltraServer istanza.
- EKS Auto Mode e Karpenter iniziano preventivamente a svuotare i nodi in un Capacity Block 10 minuti prima dell'inizio della chiusura di EC2, in modo che i carichi di lavoro abbiano il tempo di effettuare un checkpoint e spegnersi correttamente.

## Capacità statica NodePools

EKS Auto Mode e Karpenter supportano la capacità  statica  NodePools, che mantiene un numero fisso di nodi indipendentemente dalla richiesta del carico di lavoro. I pool statici eliminano i ritardi dovuti all'avvio a freddo per le inferenze sensibili alla latenza e consentono di riservare un ingombro minimo all'infrastruttura del cluster.

La capacità statica viene configurata impostando il campo su. `replicas` NodePool

Considerazioni

- Una volta `replicas` impostata su a NodePool, non è possibile rimuoverla. Non è NodePool possibile passare dalla fornitura di capacità statica a quella dinamica e viceversa.
- La capacità statica non NodePools viene considerata per il consolidamento. Impostata `limits.nodes` sopra `replicas` per consentire il ridimensionamento temporaneo durante la deriva o la scadenza dell'AMI.
- Per una distribuzione prevedibile delle zone di disponibilità (AZ), create una capacità statica NodePool per AZ anziché estenderla su più zone in un unico pool.

EKS Auto Mode

L'esempio seguente mostra una capacità statica NodePool che utilizza la modalità automatica EKS predefinita NodeClass e ne crea una statica NodePool con 4 nodi (`replicas`) che può contenere al massimo 6 nodi (). `limits.nodes`

```
apiVersion: karpenter.sh/v1
kind: NodePool
```

```
metadata:
  name: gpu-static-inference
spec:
  replicas: 4
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values: ["g6e"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-east-1a"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
  limits:
    nodes: 6   # Allow temporary headroom during node replacement
```

Self-managed Karpenter

Con Karpenter autogestito, la capacità statica è limitata dalla `StaticCapacity` funzionalità alpha (lanciata nella versione Karpenter v1.8), che deve essere abilitata nei valori Helm:

```
settings:
  featureGates:
    staticCapacity: true
```

Fa NodePool riferimento a un `EC2NodeClass` nome personalizzato `my-nodeclass` e crea uno statico NodePool con 4 nodi (`replicas`) che possono essere al massimo 6 nodi (). `limits.nodes`

```
apiVersion: karpenter.sh/v1
kind: NodePool
```

```
metadata:
  name: gpu-static-inference
spec:
  replicas: 4
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: my-nodeclass
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "karpenter.k8s.aws/instance-family"
          operator: In
          values: ["g6e"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-east-1a"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
  limits:
    nodes: 6   # Allow temporary headroom during node replacement
```

## Blocchi di capacità per ML

I Capacity Blocks for ML consentono di P-family riservare istanze di Trainium per una finestra futura definita. Sono prepagate, quindi EKS Auto Mode e Karpenter le modellano come gratuite e danno loro priorità rispetto a Spot. On-Demand I Capacity Blocks for ML possono avere una durata di prenotazione di 1-14 giorni o multipli di 7 giorni, fino a 182 giorni (6 mesi).

Per utilizzare Capacity Blocks for ML con EKS Auto Mode o Karpenter, configuralo inserendo l'ID `capacityReservationSelectorTerms` di prenotazione della capacità nel tuo. NodeClass Non è possibile utilizzare la corrispondenza delle prenotazioni aperte con Capacity Blocks for ML. Un termine può specificare un ID, un set di tag o criteri di corrispondenza delle istanze tra cui effettuare la selezione. Quando si specificano i tag, selezionerà tutte le prenotazioni di capacità accessibili dall'account con i tag corrispondenti. Questo può essere ulteriormente limitato specificando l'ID dell'account del proprietario.

Per altri esempi, consulta la documentazione di Karpenter.

EKS Auto Mode

Creane una `NodeClass` che faccia riferimento alla tua prenotazione Capacity Block, quindi creane una NodePool che la utilizzi.

Con `consolidateAfter: Never` set, Karpenter non tenterà di sostituire, unire o terminare i nodi per ridurre i costi o impacchettare i carichi di lavoro in modo più efficiente. Questa opzione è consigliata per i Capacity Blocks perché la capacità è già prepagata.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: capacity-block-gpu
spec:
  capacityReservationSelectorTerms:
    - id: "cr-0123456789abcdef0"   # Your Capacity Block reservation ID
    # Alternative: select by tags
    # - tags:
    #     role: "production-inference"
    #   owner: "012345678901"
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-capacity-block
spec:
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: Never
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: capacity-block-gpu
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["reserved"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
```

```
          values: ["p5", "p5e", "p5en", "p4d"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
```

Self-managed Karpenter

Creane uno `EC2NodeClass` che includa anche selettori di AMI, subnet e gruppi di sicurezza`capacityReservationSelectorTerms`, quindi creane uno NodePool che lo utilizzi.

Con `consolidateAfter: Never` set, Karpenter non tenterà di sostituire, unire o terminare i nodi per ridurre i costi o comprimere i carichi di lavoro in modo più efficiente. Questa opzione è consigliata per i Capacity Blocks perché la capacità è già prepagata.

```
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: capacity-block-gpu
spec:
  amiSelectorTerms:
    - alias: al2023@latest
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: ml-cluster
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: ml-cluster
  capacityReservationSelectorTerms:
    - id: "cr-0123456789abcdef0" # Your Capacity Block reservation ID
    # Alternative: select by tags
    # - tags:
    #     role: "production-inference"
    #   owner: "012345678901"
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-capacity-block
spec:
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: Never
```

```
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: capacity-block-gpu
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["reserved"]
        - key: "karpenter.k8s.aws/instance-family"
          operator: In
          values: ["p5", "p5e", "p5en", "p4d"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
```

## On-Demand Prenotazioni di capacità (ODCR)

Gli ODCR garantiscono la capacità in una specifica zona di disponibilità (AZ) senza un impegno a lungo termine. La fatturazione prevede On-Demand tariffe standard indipendentemente dal fatto che la capacità venga utilizzata o meno. Gli ODCR supportano tutte le famiglie di GPU NVIDIA, incluse le G-family istanze non supportate da Capacity Blocks for ML. Gli ODCR sono prepagati, quindi EKS Auto Mode e Karpenter li modellano come gratuiti e danno loro priorità rispetto a Spot. On-Demand

Gli ODCR si comportano diversamente dai Capacity Blocks for ML al termine della prenotazione. Quando un ODCR scade o viene annullato, l'istanza continua a funzionare come standard. On-Demand Per informazioni dettagliate, vedi the section called “Comportamento alla scadenza della prenotazione”.

Per utilizzare gli ODCR con EKS Auto Mode o Karpenter, configura `capacityReservationSelectorTerms` i termini di prenotazione della capacità nel tuo. NodeClass Un termine può specificare un ID, un set di tag o criteri di corrispondenza delle istanze tra cui effettuare la selezione. Quando si specificano i tag, selezionerà tutte le prenotazioni di capacità accessibili dall'account con i tag corrispondenti. Quando specifica i criteri di corrispondenza delle istanze, seleziona le prenotazioni in base al loro comportamento di corrispondenza: aperto (corrisponde a tutte le istanze compatibili) o targetizzato (corrisponde solo alle istanze mirate esplicitamente). Questo può essere ulteriormente limitato specificando l'ID dell'account del proprietario.

Per altri esempi, consulta la documentazione di Karpenter.

EKS Auto Mode

Crea un `NodeClass` con `capacityReservationSelectorTerms` e un NodePool che dia priorità `reserved` con fallback. `on-demand` Aggiungi `topology.kubernetes.io/zone` all'AZ dell'ODCR:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: odcr-gpu-production
spec:
  capacityReservationSelectorTerms:
    - id: "cr-0987654321fedcba0"
    # Alternative: select by tags
    # - tags:
    #     Purpose: "production-inference"
    #   owner: "012345678901"
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-reserved-production
spec:
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: Never
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: odcr-gpu-production
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["reserved", "on-demand"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values: ["p5", "g6e"]
        - key: "topology.kubernetes.io/zone"
          operator: In
```

```
          values: ["us-east-1a"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
```

Self-managed Karpenter

Crea un selettore `EC2NodeClass` con AMI, subnet e gruppo di sicurezza oltre `acapacityReservationSelectorTerms`, quindi crea: NodePool

```
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: odcr-gpu-production
spec:
  amiSelectorTerms:
    - alias: al2023@latest
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: ml-cluster
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: ml-cluster
  capacityReservationSelectorTerms:
    - id: "cr-0987654321fedcba0"
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-reserved-production
spec:
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: Never
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: odcr-gpu-production
      requirements:
        - key: "karpenter.sh/capacity-type"
```

```
          operator: In
          values: ["reserved", "on-demand"]
        - key: "karpenter.k8s.aws/instance-family"
          operator: In
          values: ["p5", "g6e"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-east-1a"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
```

## On-Demand

On-Demand è il tipo di capacità predefinito e può essere utilizzato con il provisioning statico o dinamico in EKS Auto Mode e Karpenter. Puoi richiedere esplicitamente le On-Demand istanze impostando il tuo. `karpenter.sh/capacity-type: on-demand` NodePool EKS Auto Mode e Karpenter selezionano l'istanza con il prezzo più basso che soddisfa le richieste di risorse del Pod. Utilizzalo On-Demand per lo sviluppo, la prototipazione, il ridimensionamento imprevedibile delle inferenze e qualsiasi carico di lavoro che richieda una disponibilità immediata senza rischi di interruzione.

### EKS Auto Mode

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-ondemand
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "eks.amazonaws.com/instance-family"
```

```
          operator: In
          values: ["g6", "g6e", "g7e"]
        - key: "eks.amazonaws.com/instance-gpu-manufacturer"
          operator: In
          values: ["nvidia"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
```

Self-managed Karpenter

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-ondemand
spec:
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "karpenter.k8s.aws/instance-family"
          operator: In
          values: ["g6", "g6e", "g7e"]
        - key: "karpenter.k8s.aws/instance-gpu-manufacturer"
          operator: In
          values: ["nvidia"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
```

## Spot

Spot offre un risparmio fino al 90% rispetto all'utilizzo di capacità EC2 di riserva. On-Demand AWS può recuperare le istanze Spot con un preavviso di interruzione di 2 minuti. Massimizza la

disponibilità elencando più famiglie di istanze su. NodePool Associa i carichi di lavoro Spot a un checkpoint `PodDisruptionBudget` and allo storage durevole (Amazon S3 o Amazon EFS) a intervalli regolari in modo che i Pods possano salvare lo stato durante la finestra di esaurimento.

Spot è la soluzione ideale per carichi di lavoro di formazione e inferenza riprendibili e tolleranti ai guasti, in cui un'interruzione occasionale è accettabile in cambio di significativi risparmi sui costi.

I candidati più comuni includono:

- Ottimizzazione e scansione degli iperparametri: molte brevi prove parallele che possono essere ritentate se interrotte.
- Formazione distribuita con checkpoint: processi di lunga durata che salvano periodicamente lo stato in S3 o FSx e possono riprendere dall'ultimo checkpoint dopo la perdita del nodo.
- Inferenza batch e offline: assegnazione di punteggi su larga scala rispetto a set di dati in cui la latenza end-to-end viene misurata in ore, non in secondi.
- Pipeline di preelaborazione dei dati e ingegneria delle funzionalità: trasformazioni parallele su set di dati di grandi dimensioni.
- Valutazione e benchmarking dei modelli: lavori ripetibili che producono risultati idempotenti.
- Sviluppo, prototipazione e notebook: sperimentazione interattiva in cui gli utenti possono tollerare riavvii occasionali.

Evita Spot per inferenze in tempo reale sensibili alla latenza, endpoint di SLA-bound produzione e carichi di lavoro che non superano il checkpoint o non possono tollerare i riavvii.

Puoi richiedere esplicitamente le istanze Spot impostando il tuo. `karpenter.sh/capacity-type: spot` NodePool

EKS Auto Mode

La modalità automatica EKS gestisce le interruzioni Spot in modo nativo. Non è richiesta alcuna coda SQS o Node Termination Handler.

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-spot
spec:
  disruption:
    budgets:
```

```
      - nodes: 10%
    consolidationPolicy: WhenEmpty
    consolidateAfter: 1h
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["spot"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values: ["g6", "g6e", "g7e"]
        - key: "eks.amazonaws.com/instance-gpu-manufacturer"
          operator: In
          values: ["nvidia"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
  limits:
    resources:
      nvidia.com/gpu: "64"
```

Self-managed Karpenter

Self-managed Karpenter richiede di abilitare la gestione nativa delle interruzioni sul controller Karpenter (non sul NodePool) configurando una coda di interruzione: una coda SQS che riceve gli eventi EC2 Spot interruption e Rebalance Recommendation. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/rebalance-recommendations.html Questa operazione viene configurata una volta al momento dell'installazione.

Se installi Karpenter direttamente con Helm, imposta`settings.interruptionQueue`: `values.yaml`

```
# karpenter values.yaml (Helm)
settings:
  clusterName: my-cluster
  interruptionQueue: my-queue   # Name of the SQS queue receiving Spot events
```

Se avvii Karpenter con`eksctl`, `withSpotInterruptionQueue: true` impostalo nel file di configurazione del cluster. `eksctl`crea la coda e le EventBridge regole SQS e configura il controller Karpenter per utilizzarle.

```
# eksctl ClusterConfig
karpenter:
  version: "${KARPENTER_VERSION}"
  withSpotInterruptionQueue: true
```

Una volta che il controller è impostato per utilizzare la coda, non è necessaria alcuna configurazione aggiuntiva sulle singole risorse. NodePool La gestione delle interruzioni si applica a tutto il cluster.

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-spot
spec:
  disruption:
    budgets:
      - nodes: 10%
    consolidationPolicy: WhenEmpty
    consolidateAfter: 1h
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["spot"]
        - key: "karpenter.k8s.aws/instance-family"
          operator: In
          values: ["g6", "g6e", "g7e"]
        - key: "karpenter.k8s.aws/instance-gpu-manufacturer"
          operator: In
          values: ["nvidia"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
```

```
        effect: NoSchedule
limits:
  resources:
    nvidia.com/gpu: "64"
```

# Gestisci l'elaborazione per i AI/ML carichi di lavoro su Amazon EKS con gruppi di nodi

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Questa sezione spiega come gestire l'elaborazione accelerata (AWS Trainium, GPU NVIDIA) per i carichi di lavoro di addestramento e inferenza AI utilizzando gruppi di nodi gestiti Amazon EKS o nodi autogestiti.

I gruppi di nodi gestiti da EKS e i nodi autogestiti utilizzano EC2 Auto Scaling Groups (ASG). I gruppi di nodi gestiti EKS dispongono di API EKS dedicate per la creazione, l'aggiornamento e l'eliminazione dei nodi e dispongono anche di funzionalità di riparazione dei nodi e hook di terminazione del ciclo di vita integrati. I nodi autogestiti EKS vengono distribuiti e gestiti direttamente tramite le API EC2.

Con queste opzioni, puoi definire in anticipo il tipo di istanza, il conteggio desiderato, i limiti di scalabilità e il modello di lancio di EC2. Prendi in considerazione l'utilizzo di gruppi di nodi gestiti EKS o nodi autogestiti se disponi anche di carichi di lavoro non EKS e preferisci la coerenza della configurazione attraverso i modelli di lancio di EC2. I gruppi di nodi EKS sono ideali per la formazione e l'ottimizzazione di carichi di lavoro in cui l'impronta di elaborazione accelerata è nota in anticipo. Nota, sia EKS Auto Mode che Karpenter supportano anche il provisioning statico della capacità, vedi per ulteriori informazioni. the section called “EKS Auto Mode e Karpenter”

I gruppi di nodi gestiti EKS e i nodi autogestiti supportano tutte le opzioni di acquisto di elaborazione accelerata (SpotOn-Demand, Capacity Reservations, On-Demand Capacity Blocks for ML). È possibile creare un gruppo di nodi gestito o autogestito separato per tipo di capacità, ciascuno con il proprio modello di lancio, tipi di istanza e configurazione di scalabilità. In questo modo è possibile ASG-backed controllare in modo esplicito ogni pool di capacità senza una logica di provisioning dinamica eterogenea.

## Gruppi di nodi gestiti da EKS e nodi autogestiti

La scelta tra gruppi di nodi gestiti EKS e nodi autogestiti dipende dal livello di personalizzazione e controllo richiesto. I gruppi di nodi gestiti EKS consentono la personalizzazione di un sottoinsieme del modello di lancio EC2, mentre i nodi autogestiti supportano l'intera gamma del modello di lancio EC2. Se non hai un motivo specifico per personalizzare e gestire autonomamente il ciclo di vita dei nodi, inizia con i gruppi di nodi gestiti da EKS e passa ai nodi autogestiti solo quando un requisito specifico lo impone.

Utilizza i gruppi di nodi gestiti quando: desideri che EKS gestisca la selezione degli AMI, il bootstrap dei nodi, gli aggiornamenti continui, la riparazione dei nodi e i flussi di lavoro Graceful Drain per tuo conto. I gruppi di nodi gestiti EKS sono il punto di partenza consigliato se non preferite utilizzare EKS Auto Mode o Karpenter per carichi di lavoro di addestramento e inferenza. Quando si utilizza Capacity Blocks for ML, i gruppi di nodi gestiti da EKS creano automaticamente una politica di scalabilità pianificata che prosciuga il gruppo di nodi 40 minuti prima del termine della prenotazione, eliminando la necessità di utilizzare il AWS Node Termination Handler o la propria automazione scale-down. Utilizza i gruppi di nodi gestiti EKS quando utilizzi un' EKS-optimized AMI supportata, quando non hai bisogno di personalizzazioni approfondite dei modelli di avvio EC2 a livello di kernel e quando desideri un percorso di aggiornamento dei nodi più semplice per le versioni di Kubernetes.

Utilizza i gruppi di nodi autogestiti quando: hai bisogno del pieno controllo sul modello di lancio EC2, sull'AMI, sui parametri del kernel, sulla configurazione di runtime del contenitore o sugli script di bootstrap personalizzati. Gli scenari ML più comuni includono l'ottimizzazione delle impostazioni del kernel e della NIC per l'addestramento distribuito con Elastic Fabric Adapter (EFA) o l'integrazione con un controller del ciclo di vita dei nodi personalizzato. Self-managed i nodi ti offrono la flessibilità necessaria per fornire tutti i dati utente e il profilo di istanza IAM di cui hai bisogno, ma ti assumi la responsabilità degli aggiornamenti, delle politiche di scalabilità pianificate e degli hook del ciclo di vita come il Node Termination Handler.AWS

## Prenota GPU con blocchi di capacità per ML

I Capacity Blocks for Machine Learning (ML) consentono di riservare istanze GPU in date future per carichi di lavoro di formazione o inferenza con limiti di tempo. Per ulteriori informazioni, consulta Capacity Blocks for ML nella Guida per l'utente di Amazon EC2.

Puoi utilizzare le prenotazioni di Capacity Block tramite gruppi di nodi gestiti EKS e nodi autogestiti. La configurazione del modello di lancio EC2 è la stessa in entrambi i casi. Il flusso di lavoro per la creazione dei nodi, il comportamento di ridimensionamento e gli hook del ciclo di vita per la terminazione del carico di lavoro differiscono tra le opzioni di provisioning.

## Considerazioni

- I blocchi di capacità sono disponibili solo per determinati tipi di istanze e AWS regioni Amazon EC2. Per ulteriori informazioni, consulta Work with Capacity Blocks Prerequisiti.
- I blocchi di capacità sono zonali. Durante la creazione del gruppo di nodi, è necessario utilizzare la sottorete nella stessa zona di disponibilità (AZ) della prenotazione del blocco di capacità.
- Se crei un gruppo di nodi prima che la prenotazione del Capacity Block diventi attiva, imposta la capacità desiderata su `0` durante la creazione del gruppo di nodi.
- Per dedicare del tempo a un corretto esaurimento del carico di lavoro, pianifica la scalabilità a zero più di 30 minuti prima del termine della prenotazione di Capacity Block. EC2 inizia a chiudere le istanze 30 minuti prima del termine della prenotazione.

## Crea gruppi di nodi con Capacity Blocks for ML

I gruppi di nodi gestiti da EKS e i nodi autogestiti richiedono l'utilizzo di un modello di lancio EC2 personalizzato destinato alla prenotazione Capacity Block. Di seguito sono riportati i campi minimi obbligatori per i gruppi di nodi gestiti da EKS e i nodi autogestiti. I campi aggiuntivi sono obbligatori per i nodi autogestiti, come illustrato nei passaggi relativi ai Self-managed nodi riportati di seguito.

`LaunchTemplateData`Devono includere:

- `InstanceMarketOptions` con `MarketType` impostato su `"capacity-block"`
- `CapacityReservationSpecification: CapacityReservationTarget`con `CapacityReservationId` impostato sul Capacity Block ID. Ad esempio, *`cr-0123456789abcdef0`*.
- `InstanceType`impostato sul tipo di istanza della prenotazione Capacity Block. Ad esempio, *`p5.48xlarge`*.

Questi requisiti sono illustrati negli esempi seguenti per la creazione del modello di lancio per i gruppi di nodi gestiti da EKS e i nodi autogestiti.

Managed node groups

1. Crea un file denominato `eks-capacity-block-lt.json` con i seguenti contenuti.

   Sostituisci i contenuti per `CapacityReservationId` e `InstanceType` con i valori del tuo Capacity Block. Per ulteriori informazioni sui campi aggiuntivi del modello di lancio EC2,

consulta the section called "Modelli di avvio" e Usa i blocchi di capacità per carichi di lavoro di machine learning.

```
{
  "LaunchTemplateData": {
    "InstanceMarketOptions": {
      "MarketType": "capacity-block"
    },
    "CapacityReservationSpecification": {
      "CapacityReservationTarget": {
        "CapacityReservationId": "cr-0123456789abcdef0"
      }
    },
    "InstanceType": "p5.48xlarge"
  }
}
```

2. Creazione del modello di avvio.

```
aws ec2 create-launch-template \
    --launch-template-name EKS-Capacity-Block-Launch-Template \
    --launch-template-data file://eks-capacity-block-lt.json
```

3. Utilizza il modello di lancio per creare un gruppo di nodi gestiti da EKS. Sostituisci i segnaposto nel comando seguente con i valori applicabili al tuo ambiente. Il comando seguente imposta `--ami-type` le AMI EKS-optimized NVIDIA AL2023. the section called "Usa le AMI GPU EKS Linux"Per ulteriori informazioni sulle AMI disponibili, consulta. EKS-optimized Se utilizzi un'AMI personalizzata con gruppi di nodi gestiti da EKS, specifica il tuo ID AMI nel modello di avvio.

   Quando crei un gruppo di nodi gestiti EKS che utilizza Capacity Blocks, procedi come segue:

   - Imposta `--capacity-type` su `"CAPACITY_BLOCK"`.
   - Specificate solo la sottorete nella stessa zona di disponibilità della prenotazione della capacità.
   - Se si specifica un valore diverso da zero `desiredSize` prima che la prenotazione sia attiva, il gruppo Auto Scaling segnala gli errori di avvio fino a quando la prenotazione non diventa attiva. Una volta attive, le istanze vengono avviate e l'ASG si adatta a quanto richiesto. `desiredSize`

   ```
   aws eks create-nodegroup \
       --cluster-name my-eks-cluster \
   ```

```
    --nodegroup-name eks-cb-nodes \
    --node-role "arn:aws: iam::111122223333:role/myNodeRole" \
    --region region-code \
    --subnets subnet-ExampleID1 \
    --ami-type "AL2023_x86_64_NVIDIA" \
    --scaling-config minSize=0,maxSize=2,desiredSize=0 \
    --capacity-type "CAPACITY_BLOCK" \
    --launch-template name="EKS-Capacity-Block-Launch-Template"
```

4. Se lo `desiredSize` imposti `0` al momento della creazione, ingrandisci il gruppo di nodi quando la prenotazione diventa attiva utilizzando uno dei seguenti metodi:
   - Una politica di scalabilità pianificata sull'ASG allineata all'ora di inizio della prenotazione. Per ulteriori informazioni, consulta Dimensionamento programmato per Amazon EC2 Auto Scaling nella Guida per l'utente di Amazon EC2 Auto Scaling.
   - La console Amazon EKS o `aws eks update-nodegroup-config` per aggiornare la configurazione di scalabilità.
5. Verifica che i nodi si uniscano al cluster dopo la scalabilità.
6. EKS crea automaticamente una politica di scalabilità pianificata denominata Amazon EKS Node Group Capacity Scaledown Before Reservation End per ridimensionare il gruppo di nodi fino a **`0`** 40 minuti prima della fine della prenotazione. In questo modo i Pods hanno il tempo di svuotarsi correttamente prima che EC2 inizi a terminare le istanze dopo 30 minuti. Non modificare o eliminare questa azione pianificata.

Self-managed nodes

1. Crea un file denominato `eks-capacity-block-lt.json` con i seguenti contenuti.

   Sostituisci i contenuti per `CapacityReservationId` e `InstanceType` con i valori del tuo Capacity Block. Per ulteriori informazioni sui campi aggiuntivi del modello di lancio EC2, consulta the section called "Modelli di avvio" e Usa i blocchi di capacità per carichi di lavoro di machine learning.

   Sostituisci i contenuti di `IamInstanceProfileImageId`,, `SecurityGroupIdsUserData`, `KeyName` con i valori del tuo ambiente.

```
{
  "LaunchTemplateData": {
    "InstanceMarketOptions": {
      "MarketType": "capacity-block"
```

```
    },
    "CapacityReservationSpecification": {
      "CapacityReservationTarget": {
        "CapacityReservationId": "cr-0123456789abcdef0"
      }
    },
    "IamInstanceProfile": {
      "Arn": "arn:aws: iam::111122223333:role/myNodeRole"
    },
    "ImageId": "image-id",
    "InstanceType": "p5.48xlarge",
    "KeyName": "key-name",
    "SecurityGroupIds": "sg-05b1d815d1EXAMPLE"
    ],
    "UserData": "user-data"
  }
}
```

2. Creazione del modello di avvio.

```
aws ec2 create-launch-template \
    --launch-template-name EKS-Capacity-Block-Launch-Template \
    --launch-template-data file://eks-capacity-block-lt.json
```

3. Utilizza il modello di avvio per creare l'Auto Scaling Group seguendo i passaggi seguenti. the section called "Amazon Linux" Se la prenotazione non è ancora attiva, imposta su`DesiredCapacity`. `0` Specificate solo la sottorete nella zona di disponibilità in cui è riservata la capacità.
4. Dopo aver creato i nodi autogestiti con `DesiredCapacity` set to`0`, crea una politica di scalabilità pianificata sul gruppo Auto Scaling allineata ai tempi di prenotazione del Capacity Block. Per ulteriori informazioni, consulta Dimensionamento programmato per Dimensionamento automatico Amazon EC2.

   Puoi utilizzare le istanze riservate fino a 30 minuti prima dell'orario di fine della prenotazione. Pianifica la scalabilità a zero più di 30 minuti prima dell'orario di fine in modo che i Pod abbiano il tempo di esaurirsi.

   Se preferisci la scalabilità manuale, aggiorna la capacità desiderata dell'ASG all'ora di inizio della prenotazione e di nuovo più di 30 minuti prima dell'orario di fine.
5. Per drenare correttamente i Pod, configura il Node Termination Handler. AWS Controlla gli eventi del ciclo di vita di ASG con scalabilità in base ad Amazon EC2 Auto Scaling e

consente al piano di controllo di Kubernetes di agire prima che EventBridge l'istanza diventi non disponibile. Senza di esso, gli oggetti Pods e Kubernetes possono rimanere bloccati in uno stato in sospeso. Per ulteriori informazioni, consulta AWS Node Termination Handler su. GitHub

Se non configuri il Node Termination Handler, scarica manualmente i Pod prima della finestra di 30 minuti in modo che abbiano il tempo di essere svuotati correttamente.

## Utilizzo P6e-GB200 UltraServers con Amazon EKS

Questo argomento descrive come configurare e utilizzare Amazon EKS con P6e-GB200 UltraServers. Il tipo di `p6e-gb200.36xlarge` istanza con 4 GPU NVIDIA Blackwell è disponibile solo come. P6e-GB200 UltraServers Esistono due tipi di. P6e-GB200 UltraServers `u-p6e-gb200x36` UltraServer Ha 9 `p6e-gb200.36xlarge` istanze e `u-p6e-gb200x72` UltraServer ha 18 `p6e-gb200.36xlarge` istanze.

Per ulteriori informazioni, consulta la pagina Web di Amazon EC2 P6e-GB200 UltraServers .

### Considerazioni

- Amazon EKS supporta P6e-GB200 UltraServers le versioni 1.33 e successive di Kubernetes. Questa versione di Kubernetes fornisce il supporto per Dynamic Resource Allocation (DRA), abilitato di default in EKS e nelle AMI accelerate AL2023. EKS-optimized Il DRA è un requisito per l'utilizzo di con EKS. P6e-GB200 UltraServers Il DRA non è supportato in Karpenter o EKS Auto Mode e si consiglia di utilizzare i gruppi di nodi autogestiti EKS o i gruppi di nodi gestiti da EKS quando si utilizza con EKS. P6e-GB200 UltraServers
- P6e-GB200 UltraServers sono disponibili tramite EC2 Capacity Blocks for ML. the section called "Gestione della capacità"Per informazioni su come avviare nodi EKS con Capacity Blocks, consulta la pagina.
- Quando si utilizzano gruppi di nodi gestiti EKS con Capacity Blocks, è necessario utilizzare modelli di avvio personalizzati. Quando si aggiornano i gruppi di nodi gestiti da EKS con P6e-GB200 UltraServers, è necessario impostare la dimensione desiderata del gruppo di nodi `0` prima dell'aggiornamento.
- Si consiglia di utilizzare la variante ARM NVIDIA AL2023 delle EKS-optimized AMI accelerate. Questa AMI include i componenti del nodo e la configurazione necessari con cui lavorare P6e-GB200 UltraServers. Se decidi di creare la tua AMI, sei responsabile dell'installazione e della

convalida della compatibilità del nodo e del software di sistema, inclusi i driver. Per ulteriori informazioni, consulta the section called "Usa le AMI GPU EKS Linux".

- Si consiglia di utilizzare la versione EKS-optimized AMI `v20251103` o successiva, che include la versione 580 del driver NVIDIA. Questa versione del driver NVIDIA consente a Coherent Driver-Based Memory (CDMM) di risolvere il problema della potenziale segnalazione di un eccesso di memoria. Quando CDMM è abilitato, le seguenti funzionalità non sono supportate: GPU NVIDIA (MIG) e Multi-Instance vGPU. Per ulteriori informazioni su CDMM, consulta NVIDIA Coherent Memory Management (CDMM). Driver-based
- Quando si utilizza l'operatore GPU NVIDIA con l'AMI NVIDIA EKS-optimized AL2023, è necessario disabilitare l'installazione da parte dell'operatore del driver e del toolkit, poiché questi sono già inclusi nell'AMI. Le AMI NVIDIA EKS-optimized AL2023 non includono il plug-in per dispositivi NVIDIA Kubernetes o il driver NVIDIA DRA e devono essere installate separatamente.
- Ogni `p6e-gb200.36xlarge` istanza può essere configurata con un massimo di 17 schede di rete e può sfruttare EFA per la comunicazione tra. UltraServers Il traffico di rete dei carichi di lavoro può UltraServers intersecarsi, ma per ottenere prestazioni ottimali si consiglia di pianificare i carichi di lavoro nello stesso ambiente UltraServer sfruttando IMEX per la comunicazione intra-GPU. UltraServer Per ulteriori informazioni, consulta Configurazione EFA per le istanze. P6e-GB200
- Ogni `p6e-gb200.36xlarge` istanza dispone di 3 istanze di storage da 7,5 TB. Per impostazione predefinita, l' EKS-optimized AMI non formatta e non monta gli instance store. Lo storage temporaneo del nodo può essere condiviso tra i pod che richiedono lo storage temporaneo e le immagini dei contenitori che vengono scaricate nel nodo. Se si utilizza l' EKS-optimized AMI AL2023, questa può essere configurata come parte del bootstrap dei nodi nei dati utente impostando la politica di archiviazione locale dell'istanza su RAID0. NodeConfig Se impostata su RAID0, l'istanza archivia e configura il runtime del contenitore e il kubelet per utilizzare questo storage temporaneo.

## Componenti

I seguenti componenti sono consigliati per eseguire carichi di lavoro su EKS con. P6e-GB200 UltraServers Facoltativamente, puoi utilizzare l'operatore GPU NVIDIA per installare i componenti del nodo NVIDIA. Quando si utilizza l'operatore GPU NVIDIA con l'AMI NVIDIA EKS-optimized AL2023, è necessario disabilitare l'installazione da parte dell'operatore del driver e del toolkit, poiché questi sono già inclusi nell'AMI.

| Pila | Componente |
| --- | --- |
| EKS-optimized AMI accelerata | kernel 6.12 |
| | Driver per GPU NVIDIA |
| | Driver in modalità utente NVIDIA CUDA |
| | Toolkit per container NVIDIA |
| | NVIDIA Fabric Manager |
| | Driver NVIDIA IMEX |
| | Gestore di sottorete NVIDIA NVLink |
| | Driver EFA |
| Componenti in esecuzione sul nodo | PATTUMIERA IN PVC |
| | Driver EFA DRA o plug-in per dispositivi EFA |
| | Plugin per dispositivi NVIDIA K8s |
| | Driver NVIDIA DRA |
| | NVIDIA Node Feature Discovery (NFD) |
| | NVIDIA GPU Feature Discovery (GFD) |

I componenti del nodo nella tabella precedente svolgono le seguenti funzioni:

- VPC CNI: alloca gli IP VPC come interfaccia di rete principale per i pod in esecuzione su EKS
- Driver EFA DRA o plug-in per dispositivi EFA: alloca i dispositivi EFA come reti secondarie per i pod in esecuzione su EKS. Responsabile del traffico di rete attraverso. P6e-GB200 UltraServers Per i carichi di lavoro multinodo, il GPU-to-GPU traffico all'interno di un UltraServer può fluire su NVLink multinodo. Il driver EFA DRA è consigliato per Kubernetes 1.34 e versioni successive e consente l'allocazione basata sulla topologia e la condivisione dei dispositivi. Il plug-in del dispositivo EFA è supportato per tutte le versioni di Kubernetes. Per ulteriori informazioni, consulta the section called “Dispositivi EFA”.

- Plugin per dispositivi NVIDIA Kubernetes: alloca le GPU come dispositivi per i pod in esecuzione su EKS. Si consiglia di utilizzare il plug-in per dispositivi NVIDIA Kubernetes fino a quando la funzionalità di allocazione GPU del driver NVIDIA DRA non sarà più sperimentale. Consulta le versioni dei driver NVIDIA DRA per informazioni aggiornate.
- Driver NVIDIA DRA: abilita risorse ComputeDomain personalizzate che facilitano la creazione di domini IMEX che seguono i carichi di lavoro in esecuzione. P6e-GB200 UltraServers
  - La ComputeDomain risorsa descrive un dominio Internode Memory Exchange (IMEX). Quando i carichi di lavoro con un ResourceClaim for a ComputeDomain vengono distribuiti nel cluster, il driver NVIDIA DRA crea automaticamente un IMEX DaemonSet che viene eseguito su nodi corrispondenti e stabilisce i canali IMEX tra i nodi prima dell'avvio del carico di lavoro. Per saperne di più su IMEX, consulta la panoramica di NVIDIA IMEX per sistemi NVLink multinodo.
  - Il driver NVIDIA DRA utilizza un'etichetta ID clique (`nvidia.com/gpu.clique`) applicata da NVIDIA GFD che trasmette la conoscenza della topologia di rete e del dominio NVLink.
  - È consigliabile creare un lavoro per carico di lavoro. ComputeDomain
- NVIDIA Node Feature Discovery (NFD): dipendenza richiesta per GFD per applicare le etichette dei nodi in base agli attributi a livello di nodo rilevati.
- NVIDIA GPU Feature Discovery (GFD): applica un'etichetta topologica standard NVIDIA chiamata ai nodi. `nvidia.com/gpu.clique` I nodi all'interno dello stesso `nvidia.com/gpu.clique` hanno più nodi NVLink-reachability ed è possibile utilizzare le affinità dei pod nell'applicazione per pianificare i pod sullo stesso dominio NVLink.

## Procedura

La sezione seguente presuppone che tu abbia un cluster EKS che esegue Kubernetes versione 1.33 o successiva con uno o più gruppi di nodi con l'esecuzione P6e-GB200 UltraServers dell'AMI accelerata NVIDIA ARM AL2023 ARM. EKS-optimized Consulta i collegamenti per i passaggi preliminari the section called “Gestione della capacità” per i nodi autogestiti e i gruppi di nodi gestiti da EKS.

La procedura seguente utilizza i componenti riportati di seguito.

| Nome | Versione | Description |
| --- | --- | --- |
| Operatore GPU NVIDIA | 25.3.4+ | Per la gestione del ciclo di vita dei plug-in richiesti come il plug-in per dispositivi NVIDIA Kubernetes e. NFD/GFD |
| Driver NVIDIA DRA | 25.8.0+ | Per la gestione di ComputeDomain CRD e domini IMEX. |
| Driver EFA DRA (DRANET) | Più recente | Per la UltraServer comunicazione incrociata con allocazione basata sulla topologia. Consigliato per Kubernetes 1.34+. |
| Plugin per dispositivi EFA | 0.5.14 e versioni successive | Per la comunicazione incrociata. UltraServer Supportato per tutte le versioni di Kubernetes. |

## Installa l'operatore GPU NVIDIA

L'operatore GPU NVIDIA semplifica la gestione dei componenti necessari per utilizzare le GPU nei cluster Kubernetes. Poiché il driver della GPU NVIDIA e il toolkit contenitore sono installati come parte dell' EKS-optimized AMI accelerata, questi devono essere impostati `false` nella configurazione dei valori Helm.

1. Crea un file di valori Helm denominato con la seguente configurazione. `gpu-operator-values.yaml`

```
devicePlugin:
  enabled: true
nfd:
  enabled: true
gfd:
  enabled: true
driver:
  enabled: false
toolkit:
  enabled: false
migManager:
  enabled: false
```

2. Installa l'operatore GPU NVIDIA per il tuo cluster utilizzando il `gpu-operator-values.yaml` file creato nel passaggio precedente.

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
helm repo update
```

```
helm install gpu-operator nvidia/gpu-operator \
 --namespace gpu-operator \
 --create-namespace \
 --version v25.3.4 \
 --values gpu-operator-values.yaml
```

## Installa il driver NVIDIA DRA

A partire dalla versione NVIDIA GPU Operator`v25.3.4`, il driver NVIDIA DRA deve essere installato separatamente. Si consiglia di tenere traccia delle note di rilascio dell'operatore della GPU NVIDIA poiché potrebbero cambiare in una versione futura.

1. Crea un file di valori Helm denominato `dra-values.yaml` con la seguente configurazione. Nota `tolerations` che `nodeAffinity` e che configurano il driver DRA per la distribuzione solo su nodi con una GPU NVIDIA.

```
resources:
  gpus:
    enabled: false # set to false to disable experimental gpu support
  computeDomains:
    enabled: true

controller:
  nodeSelector: null
  affinity: null
  tolerations: []

kubeletPlugin:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: "nvidia.com/gpu.present"
```

```
            operator: In
            values:
            - "true"
  tolerations:
    - key: "nvidia.com/gpu"
      operator: Exists
      effect: NoSchedule
```

2. Installa il driver NVIDIA DRA per il tuo cluster utilizzando il `dra-values.yaml` file creato nel passaggio precedente.

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
helm repo update
```

```
helm install nvidia-dra-driver-gpu nvidia/nvidia-dra-driver-gpu \
  --version="25.8.0" \
  --namespace nvidia-dra-driver-gpu \
  --create-namespace \
  -f dra-values.yaml
```

3. Dopo l'installazione, il driver DRA crea `DeviceClass` risorse che consentono a Kubernetes di comprendere e allocare le `ComputeDomain` risorse, rendendo possibile la gestione IMEX per carichi di lavoro GPU distribuiti su. P6e-GB200 UltraServers

   Verifica che le risorse DRA siano disponibili con i seguenti comandi.

```
kubectl api-resources | grep resource.k8s.io
```

```
deviceclasses           resource.k8s.io/v1  false        DeviceClass
resourceclaims          resource.k8s.io/v1  true         ResourceClaim
resourceclaimtemplates  resource.k8s.io/v1  true         ResourceClaimTemplate
resourceslices          resource.k8s.io/v1  false        ResourceSlice
```

```
kubectl get deviceclasses
```

```
NAME
compute-domain-daemon.nvidia.com
compute-domain-default-channel.nvidia.com
```

## Installa EFA per la comunicazione incrociata UltraServer

Per utilizzare la comunicazione EFA tra di loro UltraServers, installa il driver EFA DRA (DRANET) o il plug-in del dispositivo EFA. P6e-GB200 le istanze possono essere configurate con un massimo di 17 schede di rete e l'NCI primario (indice 0) deve essere di tipo `interface` e supporta fino a 100 Gbps di larghezza di banda ENA. Configura le interfacce EFA ed ENA in base alle tue esigenze durante il provisioning dei nodi. Consulta la AWS documentazione sulla configurazione EFA per P6e-GB200 le istanze per maggiori dettagli sulla configurazione EFA.

> **Important**
>
> Non installate il driver EFA DRA e il plug-in del dispositivo EFA sullo stesso nodo. I due meccanismi non possono coesistere sullo stesso nodo.

Opzione 1: installare il driver EFA DRA (DRANET)

1. Create un file di valori Helm denominato `efa-values.yaml` con la seguente configurazione.

```
tolerations:
  - key: nvidia.com/gpu
    operator: Exists
    effect: NoSchedule
```

2. Aggiungi l'archivio cartografico EKS Helm e installa il driver EFA DRA.

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update
```

```
helm install aws-dranet eks/aws-dranet --namespace kube-system -f efa-values.yaml
```

3. Verificate che il DRANET sia in esecuzione. DaemonSet

```
kubectl get daemonset -n kube-system aws-dranet
```

```
NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
 AGE
aws-dranet    2         2         2       2            2           <none>
 60s
```

4. Verificate che gli `ResourceSlice` oggetti `DeviceClass` e siano disponibili.

```
kubectl get deviceclass efa.networking.k8s.aws
```

```
NAME                    AGE
efa.networking.k8s.aws  60s
```

```
kubectl get resourceslices -l resource.k8s.io/driver=dra.net
```

Per ulteriori informazioni sull'utilizzo del driver EFA DRA, inclusa l'allocazione basata sulla topologia e la condivisione dei dispositivi, consulta. the section called "Dispositivi EFA"

Opzione 2: installa il plug-in del dispositivo EFA

1. Create un file di valori Helm denominato `efa-values.yaml` con la seguente configurazione.

```
tolerations:
  - key: nvidia.com/gpu
    operator: Exists
    effect: NoSchedule
```

2. Aggiungi l'archivio cartografico EKS Helm e installa il plug-in del dispositivo EFA.

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update
```

```
helm install efa eks/aws-efa-k8s-device-plugin -n kube-system -f efa-values.yaml
```

3. Verifica che il plug-in del dispositivo EFA sia in esecuzione. DaemonSet

```
kubectl get daemonset -n kube-system aws-efa-k8s-device-plugin-daemonset
```

```
NAME                                 DESIRED   CURRENT   READY   UP-TO-DATE
 AVAILABLE   NODE SELECTOR   AGE
aws-efa-k8s-device-plugin-daemonset   2         2         2       2           2
      <none>          60s
```

4. Verifica che i tuoi nodi dispongano di dispositivi EFA allocabili. Ad esempio, se hai configurato le istanze con 1 interfaccia solo efa in ogni gruppo NCI, si prevede che vengano visualizzati 4 dispositivi EFA allocabili per nodo.

```
kubectl get nodes "-o=custom-columns=NAME:.metadata.name,EFA:.status.allocatable.vpc
\.amazonaws\.com/efa"
```

```
NAME                                           EFA
ip-192-168-11-225.us-west-2.compute.internal   4
ip-192-168-24-96.us-west-2.compute.internal    4
```

## Multi-Node Convalida IMEX su NVLink

Per un test NVLINK NCCL multinodo e altri microbenchmark, consulta il fantastico repository di formazione distribuito. GitHub I passaggi seguenti mostrano come eseguire un test NVLink multinodo con nvbandwidth.

1. Per eseguire un test della larghezza di banda multinodo su due nodi nel dominio NVL72, installa prima l'operatore MPI:

```
kubectl create -f https://github.com/kubeflow/mpi-operator/releases/download/v0.7.0/
mpi-operator.yaml
```

2. Create un file denominato `nvbandwidth-test-job.yaml` che definisca il manifesto del test. Nota l'affinità del `nvidia.com/gpu.clique` pod per pianificare i lavoratori nello stesso dominio NVLink che ha Multi-Node la raggiungibilità di NVLink. L'esempio seguente esegue un test memcpy CE Read da dispositivo a dispositivo multinodo utilizzando cu e stampa i risultati nei log. MemcpyAsync

   A partire da NVIDIA, la versione `v25.8.0` ComputeDomains del driver DRA è elastica e può essere impostata nella definizione. `.spec.numNodes` `0` ComputeDomain Consulta le ultime note di rilascio del driver NVIDIA DRA per gli aggiornamenti.

   Può essercene solo uno ComputeDomain (canale IMEX) per nodo. Non modificate l'indirizzo della `allocationMode` ComputeDomain risorsa, in quanto ciò può impedire che i Pod ComputeDomain e i Pod che vi accedono ComputeDomain vengano allocati e programmati correttamente. `All` Per ulteriori informazioni, consulta il numero #353 del driver NVIDIA DRA.

```
---
apiVersion: resource.nvidia.com/v1beta1
kind: ComputeDomain
metadata:
  name: nvbandwidth-test-compute-domain
spec:
  numNodes: 0 # This can be set to 0 from NVIDIA DRA Driver version v25.8.0+
  channel:
    resourceClaimTemplate:
      name: nvbandwidth-test-compute-domain-channel

---
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nvbandwidth-test
spec:
  slotsPerWorker: 4 # 4 GPUs per worker node
  launcherCreationPolicy: WaitForWorkersReady
  runPolicy:
    cleanPodPolicy: Running
  sshAuthMountPath: /home/mpiuser/.ssh
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
        metadata:
          labels:
            nvbandwidth-test-replica: mpi-launcher
        spec:
          affinity:
            nodeAffinity:
              requiredDuringSchedulingIgnoredDuringExecution:
                nodeSelectorTerms:
                - matchExpressions:
                  # Only schedule on NVIDIA GB200/GB300 nodes
                  - key: node.kubernetes.io/instance-type
                    operator: In
                    values:
                    - p6e-gb200.36xlarge
                    - p6e-gb300.36xlarge
          containers:
          - image: ghcr.io/nvidia/k8s-samples:nvbandwidth-v0.7-8d103163
```

```
            name: mpi-launcher
            securityContext:
              runAsUser: 1000
            command:
            - mpirun
            args:
            - --bind-to
            - core
            - --map-by
            - ppr:4:node
            - -np
            - "8"
            - --report-bindings
            - -q
            - nvbandwidth
            - -t
            - multinode_device_to_device_memcpy_read_ce
    Worker:
      replicas: 2 # 2 worker nodes
      template:
        metadata:
          labels:
            nvbandwidth-test-replica: mpi-worker
        spec:
          affinity:
            nodeAffinity:
              requiredDuringSchedulingIgnoredDuringExecution:
                nodeSelectorTerms:
                - matchExpressions:
                  # Only schedule on NVIDIA GB200/GB300 nodes
                  - key: node.kubernetes.io/instance-type
                    operator: In
                    values:
                    - p6e-gb200.36xlarge
                    - p6e-gb300.36xlarge
            podAffinity:
              requiredDuringSchedulingIgnoredDuringExecution:
              - labelSelector:
                  matchExpressions:
                  - key: nvbandwidth-test-replica
                    operator: In
                    values:
                    - mpi-worker
                topologyKey: nvidia.com/gpu.clique
```

```
          containers:
          - image: ghcr.io/nvidia/k8s-samples:nvbandwidth-v0.7-8d103163
            name: mpi-worker
            securityContext:
              runAsUser: 1000
            env:
            command:
            - /usr/sbin/sshd
            args:
            - -De
            - -f
            - /home/mpiuser/.sshd_config
            resources:
              limits:
                nvidia.com/gpu: 4  # Request 4 GPUs per worker
              claims:
              - name: compute-domain-channel # Link to IMEX channel
          resourceClaims:
          - name: compute-domain-channel
            resourceClaimTemplateName: nvbandwidth-test-compute-domain-channel
```

3. Crea ComputeDomain e avvia il processo con il seguente comando.

```
kubectl apply -f nvbandwidth-test-job.yaml
```

4. Dopo ComputeDomain la creazione, puoi vedere che il carico di lavoro ComputeDomain ha due nodi:

```
kubectl get computedomains.resource.nvidia.com -o yaml
```

```
status:
  nodes:
  - cliqueID: <ClusterUUID>.<Clique ID>
    ipAddress: <node-ip>
    name: <node-hostname>
  - cliqueID: <ClusterUUID>.<Clique ID>
    ipAddress: <node-ip>
    name: <node-hostname>
  status: Ready
```

5. Esamina i risultati del lavoro con il seguente comando.

```
kubectl logs --tail=-1 -l job-name=nvbandwidth-test-launcher
```

Un test riuscito mostra le statistiche sulla larghezza di banda GB/s per il test memcpy a più nodi. Di seguito è riportato un esempio di risultato positivo del test.

```
...
nvbandwidth Version: ...
Built from Git version: ...

MPI version: ...
CUDA Runtime Version: ...
CUDA Driver Version: ...
Driver Version: ...

Process 0 (nvbandwidth-test-worker-0): device 0: NVIDIA GB200 (...)
Process 1 (nvbandwidth-test-worker-0): device 1: NVIDIA GB200 (...)
Process 2 (nvbandwidth-test-worker-0): device 2: NVIDIA GB200 (...)
Process 3 (nvbandwidth-test-worker-0): device 3: NVIDIA GB200 (...)
Process 4 (nvbandwidth-test-worker-1): device 0: NVIDIA GB200 (...)
Process 5 (nvbandwidth-test-worker-1): device 1: NVIDIA GB200 (...)
Process 6 (nvbandwidth-test-worker-1): device 2: NVIDIA GB200 (...)
Process 7 (nvbandwidth-test-worker-1): device 3: NVIDIA GB200 (...)

Running multinode_device_to_device_memcpy_read_ce.
memcpy CE GPU(row) -> GPU(column) bandwidth (GB/s)
           0         1         2         3         4         5         6         7
 0       N/A    821.45    822.18    821.73    822.05    821.38    822.61    821.89
 1    822.34       N/A    821.67    822.12    821.94    820.87    821.53    822.08
 2    821.76    822.29       N/A    821.58    822.43    821.15    821.82    822.31
 3    822.19    821.84    822.05       N/A    821.67    821.23    820.95    822.47
 4    821.63    822.38    821.49    822.17       N/A    821.06    821.78    822.22
 5    822.08    821.52    821.89    822.35    821.27       N/A    821.64    822.13
 6    821.94    822.15    821.68    822.04    821.39    820.92       N/A    822.56
 7    822.27    821.73    822.11    821.86    822.38    821.04    821.49       N/A

SUM multinode_device_to_device_memcpy_read_ce ...

NOTE: The reported results may not reflect the full capabilities of the platform.
Performance can vary with software drivers, hardware clocks, and system topology.
```

6. Quando il test è completo, eliminalo con il seguente comando.

```
kubectl delete -f nvbandwidth-test-job.yaml
```

# Gestisci i dispositivi hardware su Amazon EKS

Amazon EKS supporta due meccanismi Kubernetes per la gestione di dispositivi hardware specializzati nei cluster EKS: Dynamic Resource Allocation (DRA) e plug-in di dispositivo. Entrambi i meccanismi consentono ai carichi di lavoro di accedere ad acceleratori hardware come GPU NVIDIA e chip AWS Trainium e a dispositivi di rete ad alte prestazioni come Elastic Fabric Adapter (EFA).

Consigliamo di utilizzare i driver DRA per le nuove implementazioni con Kubernetes versioni 1.34 e successive quando si utilizza il provisioning della capacità statica in Karpenter, gruppi di nodi gestiti da EKS o nodi autogestiti. DRA non è attualmente supportato con EKS Auto Mode. DRA offre una selezione più completa dei dispositivi, una pianificazione basata sulla topologia e funzionalità di condivisione dei dispositivi che non sono possibili con i plug-in dei dispositivi.

Fai riferimento alla documentazione di Kubernetes per l'allocazione https://kubernetes.io/docs/concepts/scheduling-eviction/dynamic-resource-allocation/ dinamica delle risorse e ai plug-in dei dispositivi per informazioni generali su queste due funzionalità di Kubernetes. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/

## Allocazione dinamica delle risorse rispetto ai plug-in dei dispositivi

I plug-in dei dispositivi Kubernetes sono stati il meccanismo principale per esporre hardware specializzato ai carichi di lavoro Kubernetes. I plug-in per dispositivi pubblicizzano i dispositivi come risorse estese (ad esempio, `nvidia.com/gpu` o `aws.amazon.com/neuroncore`) richieste dall'utente nelle richieste e nei limiti delle risorse del contenitore. Sebbene i plug-in dei dispositivi siano ampiamente supportati e utilizzati, presentano delle limitazioni:

- I dispositivi vengono richiesti come conteggi interi opachi senza filtri basati sugli attributi.
- Nessun supporto per la condivisione dei dispositivi tra contenitori o pod.
- Nessuna allocazione espressiva basata sulla topologia tra i tipi di dispositivi.
- Le estensioni di pianificazione personalizzate sono spesso necessarie per un posizionamento intelligente.

La Dynamic Resource Allocation (DRA) è una funzionalità di Kubernetes resa generalmente disponibile nella versione 1.34 di Kubernetes che risolve queste limitazioni. Con DRA, i driver

di dispositivo pubblicano ricchi attributi di dispositivo nello scheduler Kubernetes tramite oggetti. `ResourceSlice` I dispositivi richiedono l'utilizzo di oggetti `ResourceClaim` e `ResourceClaimTemplate` che fanno riferimento a categorie. `DeviceClass`

DRA consente:

- Attribute-based selezione del dispositivo mediante espressioni CEL (Common Expression Language).
- Topology-aware allocazione che garantisce che i dispositivi siano collocati nello stesso switch PCIe o nello stesso dominio NUMA.
- Condivisione dei dispositivi tra più contenitori o pod tramite riferimenti condivisi. `ResourceClaim`
- Partizionamento dinamico e condivisione delle GPU NVIDIA quando si utilizza MIG o time-slicing

## Driver DRA per Amazon EKS

I seguenti driver DRA sono comunemente usati per gestire dispositivi hardware specializzati nei cluster Amazon EKS.

Driver NVIDIA DRA

Il driver NVIDIA DRA per GPU on GitHub consente l'allocazione flessibile e la configurazione dinamica delle GPU NVIDIA. Consulta the section called “DRA e plug-in per dispositivi” per informazioni sulla gestione delle GPU con il driver NVIDIA DRA e the section called “Usa P6e-GB200 con EKS” per informazioni sull'utilizzo di carichi di lavoro NVLink (MNNVL) con istanze `ComputeDomains` EC2 Multi-Node . Grace-Blackwell

Driver EFA DRA

Il driver EFA DRA (DRANET attivo GitHub) gestisce l'allocazione dei dispositivi Elastic Fabric Adapter (EFA) con una pianificazione basata sulla topologia che accoppia le interfacce EFA con le relative GPU o dispositivi Neuron topologicamente locali e supporta la condivisione dei dispositivi tra pod. Per ulteriori informazioni, consulta the section called “Dispositivi EFA”.

Driver Neuron DRA

Il driver Neuron DRA gestisce l'allocazione dei dispositivi AWS Trainium e AWS Inferentia2 con pianificazione basata sulla topologia, allocazione dei sottoinsiemi di dispositivi connessi e configurazione logica (LNC), senza richiedere estensioni di pianificazione personalizzate. NeuronCore Per ulteriori informazioni, consulta the section called “Acceleratori Neuron”.

# Plugin di dispositivo per Amazon EKS

I seguenti plug-in di dispositivo sono comunemente usati per gestire dispositivi hardware specializzati nei cluster Amazon EKS.

Plugin per dispositivi NVIDIA

Il plug-in per dispositivi NVIDIA su GitHub pubblicizza le GPU NVIDIA come risorse `nvidia.com/gpu` estese e monitora lo stato delle GPU.

Plugin per dispositivi EFA

Il plug-in per dispositivi EFA rileva tutti i dispositivi EFA disponibili su ciascun nodo e pubblicizza i dispositivi EFA come risorse estese. `vpc.amazonaws.com/efa`

Plugin per dispositivi Neuron

Il plug-in del dispositivo Neuron espone l'hardware e le risorse estese di Neuron. `aws.amazon.com/neuroncore` `aws.amazon.com/neuron` Scopre i dispositivi Neuron disponibili su ogni nodo, li pubblicizza come risorse allocabili e ne gestisce il ciclo di vita.

## Considerazioni

Prima di utilizzare i driver DRA su Amazon EKS, esamina le seguenti considerazioni:

- DRA è disponibile su Amazon EKS con Kubernetes versione 1.33 e successive, ma è consigliato per le versioni di Kubernetes 1.34 e successive a causa di un problema relativo a Kubernetes a monte. https://github.com/kubernetes/kubernetes/issues/133920 GitHub Il piano di controllo del cluster e i nodi devono eseguire una versione di Kubernetes che supporti DRA.
- DRA non è attualmente compatibile con EKS Auto Mode.
- DRA non è attualmente compatibile con Karpenter quando si utilizza la capacità fornita dinamicamente. È necessario utilizzare il provisioning statico della capacità in Karpenter o i gruppi di nodi gestiti da EKS o i nodi autogestiti con driver DRA.
- I driver DRA e i plug-in di dispositivo per lo stesso tipo di dispositivo non devono essere eseguiti contemporaneamente sullo stesso nodo. Disinstalla il plug-in del dispositivo prima di installare il driver DRA corrispondente o distribuiscili su nodi separati. L'esecuzione del driver DRA e del plug-in del dispositivo per lo stesso dispositivo sullo stesso nodo può causare una sottoscrizione invisibile dei dispositivi hardware sottostanti.

- DRA utilizza risorse API Kubernetes diverse (,,`DeviceClass`) rispetto ai plug-in dei `ResourceClaim` dispositivi (`ResourceClaimTemplate`,). `resource.limits` `resource.requests` È possibile utilizzare DRA per gestire le risorse estese del plug-in del dispositivo senza modificare le specifiche del carico di lavoro. Per ulteriori informazioni sulle risorse estese in DRA, consulta la documentazione di Kubernetes sul sito Web di Kubernetes.
- I driver DRA per NVIDIA, EFA e Neuron sono compatibili sia con le AMI AL2023 che con le AMI Bottlerocket. EKS-optimized Se utilizzi il driver NVIDIA DRA con Bottlerocket, disabilita il plug-in del dispositivo NVIDIA incluso nelle varianti NVIDIA Bottlerocket.
- I plug-in dei dispositivi rimangono completamente supportati per tutte le versioni di Kubernetes.

## ResourceClaim DRA vs. ResourceClaimTemplate

Quando si utilizza DRA, si richiedono dispositivi tramite `ResourceClaim` o `ResourceClaimTemplate` oggetti. Questi due tipi di risorse hanno scopi diversi e hanno comportamenti diversi nel ciclo di vita.

ResourceClaim

A `ResourceClaim` è un oggetto Kubernetes denominato che crei indipendentemente da qualsiasi Pod. Lo si fa riferimento in una specifica Pod per nome utilizzando il campo. `resourceClaimName` A `ResourceClaim` ha le seguenti caratteristiche:

- Deve esistere nel cluster prima che venga creato qualsiasi Pod che vi fa riferimento. Se il reclamo non esiste, il Pod rimane in sospeso.
- Persiste finché non lo elimini esplicitamente, indipendentemente dal fatto che i Pod vi facciano riferimento.
- Più Pod possono fare riferimento allo stesso`ResourceClaim`, il che consente la condivisione dei dispositivi. Tutti i Pod che fanno riferimento alla stessa dichiarazione condividono l'accesso agli stessi dispositivi allocati e sono programmati sullo stesso nodo.

  Usa a `ResourceClaim` quando hai bisogno di più Pod per condividere l'accesso agli stessi dispositivi o quando hai bisogno che un claim esista oltre la durata di un singolo Pod.

ResourceClaimTemplate

A `ResourceClaimTemplate` definisce un modello che Kubernetes utilizza per generare automaticamente un modello univoco `ResourceClaim` per ogni Pod. Lo si fa riferimento in una specifica Pod utilizzando il campo. `resourceClaimTemplateName` Di per

`ResourceClaimTemplate` sé non è legato a nessun Pod: è un modello riutilizzabile che persiste indipendentemente. A `ResourceClaimTemplate` ha le seguenti caratteristiche:

- Kubernetes ne crea uno nuovo `ResourceClaim` per ogni Pod che fa riferimento al modello. Ogni Pod ha il proprio set separato di dispositivi.
- Ogni generatore `ResourceClaim` è legato al ciclo di vita del Pod che ne ha innescato la creazione. Quando il Pod viene eliminato, `ResourceClaim` viene eliminato anche il generatore associato. Il Pod `ResourceClaimTemplate` stesso non viene alterato e continua a generare nuove richieste per i Pod futuri.

  Usa a `ResourceClaimTemplate` quando ogni Pod di un carico di lavoro necessita di dispositivi dedicati con configurazioni simili. Ad esempio, usa un `ResourceClaimTemplate` for Pods in a Job che utilizza l'esecuzione parallela in cui ogni Pod necessita della propria GPU o dei propri dispositivi EFA.

La tabella seguente riassume le differenze tra e. `ResourceClaim` `ResourceClaimTemplate`

| Comportamento | ResourceClaim | ResourceClaimTemplate |
|---|---|---|
| Creazione | Lo crei manualmente prima che Pod vi faccia riferimento | Kubernetes genera automatic amente un reclamo per Pod |
| Ciclo di vita | Persiste finché non lo elimini | Il modello persiste finché non viene eliminato. Ogni elemento generato `ResourceClaim` è associato al Pod che ne ha attivato la creazione. |
| Condivisione del dispositivo tra pod | Supportato. Più Pod possono fare riferimento alla stessa dichiarazione. | Non supportato. Ogni Pod riceve un reclamo separato. |
| Campo delle specifiche del pod | `resourceClaimName` | `resourceClaimTempl ateName` |

Per esempi di utilizzo di `ResourceClaim` oggetti per condividere dispositivi EFA tra Pod, vedi. the section called "Condividi i dispositivi EFA tra più pod" Per esempi di utilizzo di `ResourceClaimTemplate` oggetti con allocazione basata sulla topologia, vedere. the section called "Topology-aware EFA e GPU/Neuron allocazione dei dispositivi"

## Argomenti

- the section called "Dispositivi EFA"
- the section called "Acceleratori Neuron"
- the section called "GPU NVIDIA"
- the section called "DRA e plug-in per dispositivi"
- the section called "Multi-instance GPU (MIG)"
- the section called "Time-slicing"

# Gestisci i dispositivi EFA su Amazon EKS

Elastic Fabric Adapter (EFA) è un dispositivo di rete per istanze Amazon EC2 che consente la comunicazione tra nodi ad alte prestazioni e l'RDMA (Remote Direct Memory Access) per l'intelligenza artificiale, l'apprendimento automatico e i carichi di lavoro HPC (High Performance Computing). Amazon EKS supporta due meccanismi per la gestione dei dispositivi EFA nei cluster EKS: il driver EFA Dynamic Resource Allocation (DRA) (DRANET) e il plug-in del dispositivo EFA.

Consigliamo di utilizzare i driver DRA per le nuove implementazioni con Kubernetes versioni 1.34 e successive quando si utilizza il provisioning della capacità statica in Karpenter, gruppi di nodi gestiti da EKS o nodi autogestiti. DRA non è attualmente supportato con EKS Auto Mode. Con il driver EFA DRA, è possibile configurare un'allocazione compatibile con la topologia che accoppia le interfacce EFA con le relative GPU o dispositivi Neuron topologicamente locali e condividere i dispositivi tra pod.

## Driver EFA DRA e plug-in per dispositivi EFA

| Funzionalità | Driver EFA DRA | Plugin per dispositivi EFA |
|---|---|---|
| Versione minima di Kubernetes | 1.34 | Tutte le versioni di EKS-supported Kubernetes |
| EKS Compute | Karpenter (solo capacità statica), gruppi di nodi gestiti, nodi autogestiti | EKS Auto Mode, Karpenter, gruppi di nodi gestiti, nodi autogestiti |

| Funzionalità | Driver EFA DRA | Plugin per dispositivi EFA |
|---|---|---|
| EKS-optimized AMI | AL 2023, Bottlerocket | AL 2023, Bottlerocket |
| Pubblicità sul dispositivo | Attributi avanzati tramite `ResourceSlice` oggetti tra cui tipo di dispositivo, topologia e località PCIe | Numero intero di risorse estese `vpc.amazonaws.com/efa` |
| GPU-EFA affinità | DRA-native consapevolezza della topologia | Riconoscimento automatico della topologia (solo AMI AL2023) EKS-optimized |
| Neuron-EFA affinità | DRA-native consapevolezza della topologia | Riconoscimento automatico della topologia (solo AMI AL2023) EKS-optimized |
| Condivisione dei dispositivi | Più pod possono condividere lo stesso dispositivo EFA tramite riferimenti condivisi `ResourceClaim` | Non supportato. Ogni dispositivo EFA è assegnato esclusivamente a un Pod. |

## Creazione di nodi EKS con interfacce EFA

Quando si creano nodi EKS con interfacce EFA, le interfacce EFA vengono collegate all'istanza durante il provisioning dell'istanza. È possibile personalizzare la configurazione EFA per dispositivo e utilizzare i gruppi di posizionamento con EKS Auto Mode, Karpenter, i gruppi di nodi gestiti da EKS o i gruppi di nodi autogestiti EKS. Con EKS Auto Mode, è possibile passare la configurazione per ciascuna interfaccia di rete attraverso la sezione sottostante. `NodeClass` `advancedNetworking.networkInterfaces` Con Karpenter, si passa la configurazione per ciascuna interfaccia di rete tramite. `EC2NodeClass` Con i gruppi di nodi gestiti da EKS o i nodi autogestiti, si passa la configurazione per ciascuna interfaccia di rete con modelli di avvio.

Quando si utilizza `eksctl` per il provisioning dei nodi EKS con questa `efaEnabled` impostazione, tutte le interfacce vengono configurate in base al tipo di interfaccia`EFA`, viene creato un gruppo di EFA-specific sicurezza e il plug-in del dispositivo EFA viene installato sul cluster. Se è necessario personalizzare la configurazione EFA per dispositivo durante l'utilizzo`eksctl`, si consiglia di utilizzare

il supporto di `eksctl per i modelli di avvio. https://docs.aws.amazon.com/eks/latest/eksctl/launch-template-support.html

Gli esempi seguenti mostrano come configurare `NodeClass` e lanciare modelli con interfacce EFA. Ciò è utile per personalizzare le interfacce utilizzate per il traffico EFA rispetto a quello standard. IP-based Per informazioni sul numero di interfacce EFA supportate da ciascun tipo di istanza e su come configurarle per la massima larghezza di banda di rete, consulta Massimizzare la larghezza di banda di rete per i tipi di EFA-enabled istanza nella Amazon EC2 User Guide.

## Modalità automatica di EKS

In modalità EKS Auto, si configurano le interfacce di rete EFA utilizzando il campo in (). `advancedNetworking.networkInterfaces` `NodeClass` `eks.amazonaws.com/v1` Ogni voce specifica a`networkCardIndex`, `deviceIndex` e. `interfaceType` `interfaceType`Possono essere `interface` per interfacce di rete standard o `efa-only` per interfacce EFA dedicate al traffico RDMA senza indirizzi IP assegnati.

Una volta `networkInterfaces` configurata, le istanze avviate dal `NodePool` referente `NodeClass` utilizzano questa configurazione indipendentemente dal fatto che i Pod richiedano risorse. `vpc.amazonaws.com/efa` EKS Auto Mode non allega IP, prefissi o ENI aggiuntivi dopo l'avvio dell'istanza per i nodi con configurazione statica dell'interfaccia di rete: solo le interfacce e gli IP configurati all'avvio sono disponibili per i Pods.

Questa funzionalità può essere utilizzata con pool di nodi a capacità statica per mantenere i nodi preriscaldati per carichi di lavoro di formazione e inferenza EFA-ready distribuiti.

Esempio: EKS Auto Mode NodeClass con interfacce per P5 EFA-only

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: efa-node-class
spec:
  role: MyNodeRole
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
  securityGroupSelectorTerms:
    - tags:
        Name: "efa-security-group"
```

```
  placementGroupSelector:
    name: "ml-training-pg"
  advancedNetworking:
    networkInterfaces:
    - deviceIndex: 0
      interfaceType: interface
      networkCardIndex: 0
      secondaryIPv4PrefixCount: 1
    - networkCardIndex: 0
      deviceIndex: 1
      interfaceType: efa-only
    - networkCardIndex: 1
      deviceIndex: 0
      interfaceType: efa-only
    - networkCardIndex: 2
      deviceIndex: 0
      interfaceType: efa-only
    - networkCardIndex: 3
      deviceIndex: 0
      interfaceType: efa-only
```

Per l'elenco completo dei vincoli sulla configurazione dell'interfaccia di rete statica, vedere the section called "Configurazione statica dell'interfaccia di rete" la documentazione. NodeClass

### Karpenter

Ogni voce in `networkInterfaces` specifica a `networkCardIndex`, e. `deviceIndex` `interfaceType` `interfaceType`Può essere `interface` per interfacce di rete standard o `efa-only` per interfacce EFA dedicate al traffico RDMA e a cui non sono assegnati indirizzi IP. Una volta `networkInterfaces` configurata, le istanze avviate dal `NodePool` referente `NodeClass` utilizzano questa configurazione indipendentemente dal fatto che i Pod richiedano risorse. `vpc.amazonaws.com/efa`

Quando si utilizza Karpenter senza `networkInterfaces` specificarlo`NodeClass`, le istanze create per i Pods che richiedono `vpc.amazonaws.com/efa` hanno tutte le interfacce configurate con il tipo di interfaccia. `EFA`

La `networkInterfaces` configurazione per è stata aggiunta in Karpenter v1.11. `EC2NodeClass` L'esempio seguente mostra una `EC2NodeClass` configurazione per un' P6-B200 istanza con 1 interfaccia ENA e 8 interfacce. EFA-only

Esempio Karpenter con interfacce EC2NodeClass per EFA-only P6-B200

```
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: efa-node-class
spec:
  networkInterfaces:
  - networkCardIndex: 0
    deviceIndex: 0
    interfaceType: interface
  - networkCardIndex: 0
    deviceIndex: 1
    interfaceType: efa-only
  - networkCardIndex: 1
    deviceIndex: 0
    interfaceType: efa-only
  - networkCardIndex: 2
    deviceIndex: 0
    interfaceType: efa-only
  - networkCardIndex: 3
    deviceIndex: 0
    interfaceType: efa-only
  - networkCardIndex: 4
    deviceIndex: 0
    interfaceType: efa-only
  - networkCardIndex: 5
    deviceIndex: 0
    interfaceType: efa-only
  - networkCardIndex: 6
    deviceIndex: 0
    interfaceType: efa-only
  - networkCardIndex: 7
    deviceIndex: 0
    interfaceType: efa-only
```

## Gruppi di nodi gestiti da EKS e nodi autogestiti

Con i gruppi di nodi gestiti da EKS o i nodi autogestiti, si passa alla configurazione di ciascuna interfaccia di rete con modelli di avvio.

L'esempio seguente mostra un modello di avvio configurato per un' P6-B200 istanza con 1 interfaccia ENA e 8 EFA-only interfacce. L'interfaccia di rete principale (scheda di rete 0, indice del dispositivo

0) utilizza un `interface` tipo standard per il traffico IP, mentre le interfacce aggiuntive vengono utilizzate `efa-only` per il traffico RDMA dedicato. Regola il numero di `efa-only` interfacce in base al tipo di istanza. Per il numero di interfacce EFA supportate da ciascun tipo di istanza, consulta Massimizzare la larghezza di banda di rete per i tipi di EFA-enabled istanza nella Amazon EC2 User Guide.

Esempio di modello di avvio con interfacce per EFA-only P6-B200

Sostituisci *`security-group-id`* con i tuoi valori. Il gruppo di sicurezza deve consentire tutto il traffico in entrata e in uscita da e verso se stesso per abilitare la funzionalità EFA OS-bypass . Per ulteriori informazioni, consulta la Fase 1: Preparare un gruppo EFA-enabled di sicurezza nella Amazon EC2 User Guide.

> **Important**
>
> Non specificare `SubnetId` nel modello di avvio quando si utilizzano gruppi di nodi gestiti da EKS. EKS richiede che tutte le sottoreti siano specificate tramite l'`CreateNodegroup`API e rifiuta i modelli di avvio che includono la configurazione della sottorete.

```
{
  "LaunchTemplateName": "efa-launch-template",
  "LaunchTemplateData": {
    "InstanceType": "p6-b200.48xlarge",
    "NetworkInterfaces": [
      {
        "NetworkCardIndex": 0,
        "DeviceIndex": 0,
        "InterfaceType": "interface",
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 0,
        "DeviceIndex": 1,
        "InterfaceType": "efa-only",
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 1,
        "DeviceIndex": 0,
        "InterfaceType": "efa-only",
```

```
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 2,
        "DeviceIndex": 0,
        "InterfaceType": "efa-only",
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 3,
        "DeviceIndex": 0,
        "InterfaceType": "efa-only",
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 4,
        "DeviceIndex": 0,
        "InterfaceType": "efa-only",
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 5,
        "DeviceIndex": 0,
        "InterfaceType": "efa-only",
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 6,
        "DeviceIndex": 0,
        "InterfaceType": "efa-only",
        "Groups": ["security-group-id"]
      },
      {
        "NetworkCardIndex": 7,
        "DeviceIndex": 0,
        "InterfaceType": "efa-only",
        "Groups": ["security-group-id"]
      }
    ]
  }
}
```

## Utilizzo delle AMI con EFA EKS-optimized

Le AMI EKS-optimized AL2023 e tutte le AMI Bottlerocket includono i componenti a livello di host necessari per utilizzare EFA, in particolare i componenti installati da aws-efa-installer. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/efa-start.html#efa-start-enable Le AMI EKS AL2023 e Bottlerocket non includono il driver EFA DRA o il plug-in del dispositivo EFA e questi devono essere installati separatamente sul cluster prima di  distribuire i carichi di lavoro.

## Conservazione dell'allocazione degli indirizzi IP

EFA-enabled istanze come `p5.48xlarge` e `p6-b200.48xlarge` supportano molte interfacce di rete. Per impostazione predefinita, Amazon VPC CNI alloca gli indirizzi IP su tutti gli ENI IP-enabled collegati, il che può consumare un numero elevato di indirizzi IP dalla tua sottorete anche quando tali indirizzi non vengono utilizzati attivamente dai Pods. Nelle istanze con dozzine di interfacce di rete, questo può esaurire rapidamente lo spazio IP disponibile della sottorete.

Per ridurre il consumo di indirizzi IP sui EFA-enabled nodi, configura le interfacce di rete in modo che vengano utilizzate `efa-only` per tutte le interfacce tranne quella principale. EFA-only le interfacce sono dedicate al traffico RDMA e non hanno indirizzi IP assegnati, quindi non consumano indirizzi dalla sottorete. Ad esempio, configurazioni, vedere e. the section called “Karpenter” the section called “Gruppi di nodi gestiti da EKS e nodi autogestiti” Per il layout di interfaccia consigliato per ogni tipo di istanza, consulta  Massimizzare la larghezza di banda di rete per i tipi di EFA-enabled istanza  nella Amazon EC2 User Guide.

Oltre a utilizzare le `efa-only` interfacce, puoi configurare Amazon VPC CNI per limitare il numero di indirizzi IP e ENI caldi (preallocati). Per impostazione predefinita, il VPC CNI prealloca un pool caldo di ENI e indirizzi IP per velocizzare l'avvio del Pod, ma in casi di grandi dimensioni può riservare centinaia di indirizzi IP inutilizzati. Imposta le variabili di `WARM_ENI_TARGET` ambiente `WARM_IP_TARGET` e di ambiente su `aws-node` DaemonSet per controllare il numero di indirizzi IP e ENI di riserva che il CNI mantiene. Per ulteriori informazioni su queste impostazioni, consulta le best practice CNI di  Amazon VPC.

> **Note**
>
> Le `WARM_IP_TARGET` impostazioni `WARM_ENI_TARGET` and sono a livello di cluster e si applicano a tutti i nodi gestiti da VPC CNI. Al momento non è possibile impostare valori diversi per ogni gruppo di nodi o tipo di istanza. Se hai bisogno di un controllo più granulare

di queste impostazioni, fornisci un feedback sul problema #1834 di containers-roadmap su. GitHub

## Installa il driver EFA DRA (DRANET)

Il driver EFA DRA è integrato nel progetto DRANET upstream on, che fornisce la gestione dei dispositivi di rete basata sul cloud per GitHub Kubernetes DRA. Il driver EFA DRA e DRANET sono usati in modo intercambiabile in questa documentazione e fanno riferimento allo stesso strumento.

Il driver EFA DRA pubblicizza i dispositivi EFA come oggetti con il nome del driver e il nome. `ResourceSlice` `dra.net` `DeviceClass` `efa.networking.k8s.aws` Il driver EFA DRA viene eseguito DaemonSet su ciascun nodo e rileva automaticamente i dispositivi EFA.

Prerequisiti

- Un cluster Amazon EKS che esegue Kubernetes versione 1.34 o successiva con capacità statica fornita da Karpenter, gruppi di nodi gestiti da EKS o gruppi di nodi autogestiti.
- EFA-enabled Nodi con tipi di istanza Amazon EC2. Per un elenco dei tipi di istanze supportati, consulta Tipi di istanza supportati nella Amazon EC2 User Guide.
- Nodi con componenti a livello di host installati per EFA, consulta Installare il software EFA per maggiori informazioni. Le AMI NVIDIA e Neuron EKS-optimized AL2023 e le AMI Bottlerocket includono i componenti EFA a livello di host.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurato per comunicare con il cluster, consulta per ulteriori informazioni. the section called "`Installa o aggiorna kubectl`"

Procedura

> **Important**
>
> Non installare il driver EFA DRA sui nodi in cui è in esecuzione il plug-in del dispositivo EFA. I due meccanismi non possono coesistere sullo stesso nodo. Ciò può causare una sottoscrizione eccessiva silenziosa dei dispositivi sottostanti a più pod sullo stesso nodo.

1. Aggiungi il repository di grafici EKS Helm.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Aggiorna il tuo repository Helm locale.

```
helm repo update
```

3. Installa il driver EFA DRA sul tuo cluster usando Helm. Il driver EFA DRA rileva automaticamente che è in esecuzione su istanze EC2 tramite Instance Metadata Service (IMDS) e consente il rilevamento dei dispositivi EFA. Per impostazione predefinita, il driver EFA DRA viene distribuito come nel namespace. DaemonSet `kube-system` Per i parametri configurabili, consultate Helm values.yaml nell'archivio di grafici EKS Helm attivo. GitHub

```
helm install aws-dranet eks/aws-dranet --namespace kube-system
```

4. Verificate DaemonSet che DRANET sia in esecuzione.

```
kubectl get daemonset -n kube-system aws-dranet
```

```
NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
 AGE
aws-dranet    2         2         2       2            2           <none>
 60s
```

5. Verificare che sia `DeviceClass` stato creato.

```
kubectl get deviceclass
```

```
NAME                     AGE
efa.networking.k8s.aws   60s
```

6. Verifica che `ResourceSlice` gli oggetti siano pubblicizzati per i tuoi nodi.

```
kubectl get resourceslices --field-selector spec.driver=dra.net
```

Se si verificano errori nei passaggi precedenti, è possibile controllare i log di DRANET con il seguente comando.

```
kubectl logs -n kube-system -l app=aws-dranet
```

7. Per richiedere dispositivi EFA utilizzando il driver DRA, create un `ResourceClaim` or `ResourceClaimTemplate` che faccia riferimento all'EFA `DeviceClass` e inseritelo nelle specifiche del vostro Pod. L'esempio seguente richiede un singolo dispositivo EFA.

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: single-efa-claim
spec:
  spec:
    devices:
      requests:
      - name: efa
        exactly:
          deviceClassName: efa.networking.k8s.aws
          count: 1
---
apiVersion: v1
kind: Pod
metadata:
  name: efa-workload
spec:
  containers:
  - name: app
    ...
    resources:
      claims:
      - name: efa-device
  resourceClaims:
  - name: efa-device
    resourceClaimTemplateName: single-efa-claim
```

## Topology-aware EFA e GPU/Neuron allocazione dei dispositivi

Il driver EFA DRA supporta l'allocazione basata sulla topologia che accoppia le interfacce EFA con GPU o dispositivi Neuron sulla stessa radice PCIe. Usa il vincolo per allineare le allocazioni dei dispositivi EFA e GPU `matchAttribute` o Neuron. Per utilizzare questa funzionalità, è necessario utilizzare anche i driver NVIDIA o Neuron DRA. Per ulteriori informazioni, consultare the section called “GPU NVIDIA” e the section called “Acceleratori Neuron”.

L'esempio seguente richiede 1 interfaccia EFA allineata con 1 GPU NVIDIA:

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: aligned-efa-nvidia
spec:
  spec:
    devices:
      requests:
      - name: 1-efa
        exactly:
          deviceClassName: efa.networking.k8s.aws
          count: 1
      - name: 1-gpu
        exactly:
          deviceClassName: gpu.nvidia.com
          count: 1
      constraints:
      - requests: ["1-gpu", "1-efa"]
        matchAttribute: "resource.kubernetes.io/pcieRoot"
```

L'esempio seguente richiede 4 interfacce EFA allineate con 4 dispositivi Neuron:

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: aligned-efa-neuron
spec:
  spec:
    devices:
      requests:
      - name: 4-neurons
        exactly:
          deviceClassName: neuron.aws.com
          count: 4
      - name: 4-efas
        exactly:
          deviceClassName: efa.networking.k8s.aws
          count: 4
      constraints:
      - requests: ["4-neurons", "4-efas"]
        matchAttribute: "resource.aws.com/devicegroup4_id"
```

Il numero nel nome dell'`devicegroup`attributo corrisponde al numero di dispositivi Neuron nel gruppo di topologia connesso. Ad esempio, `resource.aws.com/devicegroup1_id` identifica un singolo dispositivo Neuron, `resource.aws.com/devicegroup4_id` identifica un gruppo di 4 dispositivi connessi e `resource.aws.com/devicegroup16_id` identifica rispettivamente gruppi di 8 `resource.aws.com/devicegroup8_id` e 16 dispositivi connessi. Scegli `matchAttribute` quello che corrisponde al dispositivo nella tua richiesta `count` in modo che i dispositivi Neuron allocati e le interfacce EFA appartengano allo stesso gruppo di topologia connessa. Per ulteriori informazioni su questi attributi, consultate la documentazione del driver Neuron DRA.

È possibile utilizzarlo `allocationMode` per semplificare il modo in cui i dispositivi EFA vengono allocati a GPU o acceleratori Neuron allineati. Il `allocationMode` campo supporta due valori: `ExactCount` (impostazione predefinita) richiede un numero specifico di dispositivi specificato da e `All` richiede tutti i dispositivi `count` corrispondenti in un pool. Ad esempio, `p5.48xlarge` nelle istanze ci sono quattro dispositivi EFA che condividono la stessa radice PCIe con una GPU. Per assegnare questi gruppi di dispositivi EFA a GPU allineate, anche se non si conosce l'esatta mappatura dei dispositivi e il numero di EFA-GPU dispositivi EFA allineati, è possibile configurare la configurazione con per i dispositivi EFA. `ResourceClaimTemplate` `allocationMode: All`

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: aligned-all-efa-one-nvidia
spec:
  spec:
    devices:
      requests:
      - name: all-efas
        exactly:
          deviceClassName: efa.networking.k8s.aws
          allocationMode: All
      - name: one-gpu
        exactly:
          deviceClassName: gpu.nvidia.com
          allocationMode: ExactCount
          count: 1
      constraints:
      - requests: ["all-efas", "one-gpu"]
        matchAttribute: "resource.kubernetes.io/pcieRoot"
```

## Condividi i dispositivi EFA tra più pod

Il driver EFA DRA supporta la condivisione di dispositivi EFA tra più Pod utilizzando un. `ResourceClaim` A differenza di a`ResourceClaimTemplate`, che genera un claim separato per ogni Pod, a `ResourceClaim` è un oggetto denominato creato indipendentemente e a cui si fa riferimento da più Pod. Tutti i Pod che fanno riferimento allo stesso `ResourceClaim` condividono l'accesso agli stessi dispositivi EFA allocati e sono programmati sullo stesso nodo in cui tali dispositivi sono disponibili.

Per condividere i dispositivi EFA tra Pod, creane uno `ResourceClaim` che richieda i dispositivi EFA, quindi fai riferimento a quell'attestazione per nome nel campo di ciascun Pod utilizzando. `resourceClaims` `resourceClaimName` `ResourceClaim`Deve esistere nel cluster prima che i Pod che vi fanno riferimento vengano creati. Se un riferimento `ResourceClaim` non esiste, i Pod rimangono in sospeso fino alla creazione del claim.

L'esempio seguente crea un file `ResourceClaim` che richiede 4 dispositivi EFA e due Pod che condividono l'accesso a tali dispositivi.

1. Creazione della `ResourceClaim`.

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaim
metadata:
  name: shared-efa
spec:
  devices:
    requests:
    - name: efa
      exactly:
        deviceClassName: efa.networking.k8s.aws
        count: 4
```

2. Fai riferimento al nome `ResourceClaim` di ogni Pod che richiede l'accesso ai dispositivi EFA. Ogni Pod utilizza `resourceClaimName` per fare riferimento all'attestazione esistente anzichéresourceClaimTemplateName.

```
apiVersion: v1
kind: Pod
metadata:
  name: training-worker
spec:
```

```
  containers:
  - name: worker
    image: my-training-image
    resources:
      claims:
      - name: efa-devices
  resourceClaims:
  - name: efa-devices
    resourceClaimName: shared-efa
---
apiVersion: v1
kind: Pod
metadata:
  name: training-monitor
spec:
  containers:
  - name: monitor
    image: my-monitor-image
    resources:
      claims:
      - name: efa-devices
  resourceClaims:
  - name: efa-devices
    resourceClaimName: shared-efa
```

Entrambi i Pod fanno riferimento allo stesso `shared-efa` `ResourceClaim` e sono programmati nel nodo in cui sono allocati i dispositivi EFA. Il `ResourceClaim` ciclo di vita è indipendente dai Pod: persiste finché non li elimini, anche se tutti i Pod che vi fanno riferimento vengono rimossi.

## Installa il plug-in del dispositivo EFA Kubernetes

Il plug-in per dispositivi EFA Kubernetes pubblicizza i dispositivi EFA come risorse estese. `vpc.amazonaws.com/efa` Richiedete dispositivi EFA in container, richieste e limiti di risorse. Per una guida completa alla configurazione dell'EFA con carichi di lavoro di formazione, consulta. the section called "Configurazione dei cluster di formazione con EFA"

> **Important**
>
> Topology-aligned l'allocazione delle GPU NVIDIA o dei dispositivi Neuron con interfacce EFA avviene automaticamente quando si utilizzano le AMI accelerate AL2023. EKS-optimized Questo allineamento automatico non si verifica quando si utilizzano AMI Bottlerocket o AMI

personalizzate. EKS-optimized Se hai bisogno di un acceleratore allineato alla topologia e di un'allocazione dei dispositivi EFA con Bottlerocket o AMI personalizzate, usa il driver EFA DRA e il driver Neuron DRA corrispondente. Per utilizzare il driver NVIDIA DRA su Bottlerocket, devi prima disabilitare il plug-in del dispositivo NVIDIA fornito in bundle con le varianti NVIDIA Bottlerocket, che richiede Bottlerocket versione 1.63.0 o successiva. Per ulteriori informazioni, consultare the section called "Topology-aware EFA e GPU/Neuron allocazione dei dispositivi" e the section called "Installa il driver NVIDIA DRA".

**Important**

A partire da NVIDIA `k8s-device-plugin` v0.19.0, il flag predefinito è, il che fa sì che il plug-in del dispositivo NVIDIA monti tutti i dispositivi in contenitori che richiedono GPU. `--mofed-enabled true` `/dev/infiniband/uverbs*` Ciò è in conflitto con il plug-in del dispositivo EFA, che dovrebbe essere il componente che gestisce l'allocazione dei dispositivi EFA in. `/dev/infiniband` Se si utilizzano gruppi di nodi gestiti da EKS o nodi autogestiti con il plug-in per dispositivi NVIDIA, è necessario disabilitare esplicitamente MOFED. Per istruzioni, consulta the section called "Installa il plug-in del dispositivo NVIDIA Kubernetes". La modalità automatica EKS non abilita MOFED per impostazione predefinita e non è interessata da questo problema.

Prerequisiti

- Un cluster Amazon EKS.
- Nodi con tipi di istanza EFA-enabled Amazon EC2. Per un elenco dei tipi di istanze supportati, consulta Tipi di istanza supportati nella Amazon EC2 User Guide.
- Nodi con componenti a livello di host installati per EFA, consulta Installare il software EFA per maggiori informazioni. Le AMI EKS-optimized AL2023 e le AMI Bottlerocket includono i componenti EFA a livello di host.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurato per comunicare con il cluster, consulta per ulteriori informazioni. the section called "`Installa o aggiorna kubectl`"

Procedura

1. Aggiungi il repository di grafici EKS Helm.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Aggiorna il tuo repository Helm locale.

```
helm repo update
```

3. Installa il plug-in del dispositivo EFA.

```
helm install efa eks/aws-efa-k8s-device-plugin -n kube-system
```

4. Verifica che il plug-in del dispositivo EFA DaemonSet sia in esecuzione.

```
kubectl get daemonset -n kube-system efa-aws-efa-k8s-device-plugin
```

```
NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE
 AVAILABLE   NODE SELECTOR   AGE
efa-aws-efa-k8s-device-plugin         2         2         2       2            2
      <none>          60s
```

5. Verifica che i tuoi nodi dispongano di risorse EFA allocabili.

```
kubectl get nodes "-o=custom-columns=NAME:.metadata.name,EFA:.status.allocatable.vpc
\.amazonaws\.com/efa"
```

```
NAME                                          EFA
ip-192-168-11-225.us-west-2.compute.internal   4
ip-192-168-24-96.us-west-2.compute.internal    4
```

6. Per richiedere dispositivi EFA utilizzando il plug-in del dispositivo, specifica la `vpc.amazonaws.com/efa` risorsa nelle richieste e nei limiti delle risorse del contenitore.

```
apiVersion: v1
kind: Pod
metadata:
  name: efa-workload
spec:
  containers:
```

```
- name: app
  ...
  resources:
    limits:
      vpc.amazonaws.com/efa: 4
      hugepages-2Mi: ...
    requests:
      vpc.amazonaws.com/efa: 4
      hugepages-2Mi: ...
```

## Gestisci i dispositivi Neuron su Amazon EKS

AWS Trainium e AWS Inferentia sono chip di apprendimento automatico appositamente progettati da. AWS Amazon EKS supporta due meccanismi per la gestione dei dispositivi Neuron nei cluster EKS: il driver Neuron DRA e il plug-in per dispositivi  Neuron Kubernetes.

Consigliamo di utilizzare i driver DRA per le nuove implementazioni con Kubernetes versioni 1.34 e successive quando si utilizza il provisioning della capacità  statica in Karpenter, gruppi di nodi gestiti da EKS o nodi autogestiti.  DRA non è attualmente supportato con EKS Auto Mode. Il driver Neuron DRA fornisce l'allocazione basata sulla topologia, la pianificazione dei sottoinsiemi di dispositivi connessi, la configurazione logica NeuronCore (LNC) e l'allocazione multinodo senza richiedere estensioni di pianificazione personalizzate. UltraServer

### Driver Neuron DRA e plug-in per dispositivi Neuron

| Funzionalità | Driver Neuron DRA | Plugin per dispositivi Neuron |
|---|---|---|
| Versione minima di Kubernetes | 1.34 | Tutte le versioni di EKS-supported Kubernetes |
| EKS Compute | Karpenter (solo capacità statica), gruppi di nodi gestiti, nodi autogestiti | EKS Auto Mode, Karpenter, gruppi di nodi gestiti, nodi autogestiti |
| EKS-optimized Supporto AMI | AL 2023 (Neuron), Bottlerocket | AL 2023 (Neuron), Bottlerocket |
| Pubblicità del dispositivo | Attributi avanzati tramite `ResourceSlice` oggetti tra cui ID del dispositi | Numero intero e risorse estese `aws.amazon.com/neuron` `aws.amazon.com/neuroncore` |

| Funzionalità | Driver Neuron DRA | Plugin per dispositivi Neuron |
|---|---|---|
| | vo, tipo di istanza, topologia, versione del driver e località EFA | |
| Sottoinsiemi di dispositivi connessi | Assegna sottoinsiemi di 1, 4, 8 o 16 dispositivi Neuron connessi utilizzan do vincoli topologici | Richiede l'estensione dello scheduler Neuron per l'allocazione contigua dei dispositivi |
| Configurazione LNC | Per-workload NeuronCore Configura zione logica (LNC=1 o LNC=2) tramite parametri `ResourceC laimTemplate` | Richiede la preconfigurazione nei modelli di avvio EC2 |
| Attribute-based selezione | Filtrare i dispositivi per tipo di istanza, versione del driver e altri attributi utilizzando le espressioni CEL | Non supportata |
| Topology-aware Allocazione EFA | DRA-native consapevolezza della topologia | Riconoscimento automatico della topologia (solo AMI AL2023) EKS-optimized |

## Installa il driver Neuron DRA

Il driver Neuron DRA pubblicizza i dispositivi Neuron come oggetti con il nome. `ResourceSlice` `DeviceClass` `neuron.aws.com` Il driver funziona come un DaemonSet e rileva automaticamente i dispositivi Neuron e i relativi attributi di topologia.

Informazioni dettagliate sul driver Neuron DRA sono disponibili nella documentazione Neuron DRA. https://awsdocs-neuron.readthedocs-hosted.com/en/latest/containers/neuron-dra.html#neuron-dra-driver-attributes-reference

Prerequisiti

- Un cluster Amazon EKS che esegue Kubernetes versione 1.34 o successiva con capacità statica fornita da Karpenter, gruppi di nodi gestiti da EKS o gruppi di nodi autogestiti.
- Nodi con tipi di istanza Trainium o Inferentia2. AWS

- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurati per comunicare con il tuo cluster, vedi the section called "`Installa o aggiorna kubectl`" per maggiori informazioni.

Procedura

> **Important**
>
> Non installare il driver Neuron DRA sui nodi in cui è in esecuzione il plug-in del dispositivo Neuron. I due meccanismi non possono coesistere sullo stesso nodo. Ciò può causare una sottoscrizione eccessiva silenziosa dei dispositivi sottostanti a più pod sullo stesso nodo.

1. Installa il driver Neuron DRA usando Helm.

```
helm upgrade --install neuron-helm-chart oci://public.ecr.aws/neuron/neuron-helm-
chart \
    --namespace neuron-dra-driver \
    --create-namespace \
    --set "devicePlugin.enabled=false" \
    --set "npd.enabled=false" \
    --set "draDriver.enabled=true"
```

   Per impostazione predefinita, il driver viene distribuito come file DaemonSet nel `neuron-dra-driver` namespace con. `DeviceClass` `neuron.aws.com`

2. Verificare che il driver DRA sia in esecuzione. DaemonSet

```
kubectl get ds -n neuron-dra-driver neuron-dra-driver-kubelet-plugin
```

```
NAME                              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
  NODE SELECTOR   AGE
neuron-dra-driver-kubelet-plugin  1         1         1       1            1
  <none>          60s
```

3. Verificare che sia `DeviceClass` stato creato.

```
kubectl get deviceclass neuron.aws.com
```

```
NAME            AGE
neuron.aws.com  60s
```

4. Verifica che `ResourceSlice` gli oggetti siano pubblicizzati per i tuoi nodi.

```
kubectl get resourceslice
```

Consulta la documentazione di Neuron DRA per informazioni sugli attributi degli oggetti disponibili`ResourceSlice`.

Richiedi i dispositivi Neuron in un Pod

Per richiedere dispositivi Neuron utilizzando il driver DRA, creane uno `ResourceClaimTemplate` che faccia riferimento al `neuron.aws.com` `DeviceClass` e inseriscilo nelle specifiche del tuo Pod.

L'esempio seguente richiede tutti i dispositivi Neuron su un'istanza: `trn2.48xlarge`

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: all-neurons
spec:
  spec:
    devices:
      requests:
      - name: neurons
        exactly:
          deviceClassName: neuron.aws.com
          selectors:
          - cel:
              expression: "device.attributes['neuron.aws.com'].instanceType ==
 'trn2.48xlarge'"
          allocationMode: All
---
apiVersion: v1
kind: Pod
metadata:
  name: neuron-workload
spec:
  containers:
  - name: app
```

```
    ...
    resources:
      claims:
      - name: neurons
  resourceClaims:
  - name: neurons
    resourceClaimTemplateName: all-neurons
```

Assegna sottoinsiemi di dispositivi connessi

Il driver Neuron DRA può allocare sottoinsiemi di dispositivi Neuron connessi senza richiedere l'estensione dello scheduler Neuron. https://awsdocs-neuron.readthedocs-hosted.com/en/latest/containers/tutorials/k8s-neuron-scheduler.html Le dimensioni dei sottoinsiemi supportate sono 1, 4, 8 o 16 dispositivi. Utilizzate il `matchAttribute` vincolo con un ID di gruppo di topologia per assicurarvi che i dispositivi siano connessi.

L'esempio seguente richiede 4 dispositivi Neuron connessi:

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: 1x4-connected-neurons
spec:
  spec:
    devices:
      requests:
      - name: neurons
        exactly:
          deviceClassName: neuron.aws.com
          allocationMode: ExactCount
          count: 4
          selectors:
          - cel:
              expression: "device.attributes['neuron.aws.com'].instanceType ==
 'trn2.48xlarge'"
      constraints:
      - requests: ["neurons"]
        matchAttribute: "resource.aws.com/devicegroup4_id"
```

I `matchAttribute` valori supportati per i sottoinsiemi connessi sono`resource.aws.com/devicegroup1_id`, `resource.aws.com/devicegroup4_id`resource.aws.com/devicegroup8_id, e. `resource.aws.com/devicegroup16_id` Il numero nel nome

dell'`devicegroup`attributo corrisponde al numero di dispositivi Neuron nel gruppo di topologia connessa. Ad esempio, `resource.aws.com/devicegroup1_id` identifica un singolo dispositivo Neuron, `resource.aws.com/devicegroup4_id` identifica un gruppo di 4 dispositivi connessi e `resource.aws.com/devicegroup16_id` identifica rispettivamente gruppi di 8 `resource.aws.com/devicegroup8_id` e 16 dispositivi connessi. Scegli `matchAttribute` quello che corrisponde al dispositivo nella tua richiesta `count` in modo che i dispositivi allocati appartengano allo stesso gruppo di topologia connessa. Per ulteriori informazioni su questi attributi, consulta la documentazione del driver Neuron DRA.

Configura Logical NeuronCores (LNC)

Il driver Neuron DRA consente la configurazione logica per carico di lavoro tramite parametri. NeuronCore `ResourceClaimTemplate` Ciò elimina la necessità di preconfigurare LNC in EC2 Launch Templates.

L'esempio seguente richiede tutti i dispositivi Neuron con LNC impostato su 1:

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: all-neurons-lnc-1
spec:
  spec:
    devices:
      requests:
      - name: neurons
        exactly:
          deviceClassName: neuron.aws.com
          selectors:
          - cel:
              expression: "device.attributes['neuron.aws.com'].instanceType ==
 'trn2.48xlarge'"
          allocationMode: All
      config:
      - requests: ["neurons"]
        opaque:
          driver: neuron.aws.com
          parameters:
            apiVersion: neuron.aws.com/v1
            kind: NeuronConfig
            logicalNeuronCore: 1
```

Assegna dispositivi Neuron con interfacce EFA allineate

Per informazioni, consultare the section called "Topology-aware EFA e GPU/Neuron allocazione dei dispositivi".

## Installa il plug-in del dispositivo Neuron Kubernetes

Il plug-in per dispositivi Neuron Kubernetes pubblicizza i dispositivi Neuron come risorse estese. `aws.amazon.com/neuron` NeuronCores `aws.amazon.com/neuroncore` Richiedete i dispositivi Neuron in container, richieste e limiti di risorse.

Prerequisiti

- Un cluster Amazon EKS.
- Nodi con componenti a livello di host installati per istanze AWS Trainium o AWS Inferentia. Questi sono inclusi se si utilizzano le AMI accelerate EKS AL2023 o le AMI EKS Bottlerocket.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurato per comunicare con il cluster, vedi per ulteriori informazioni. the section called "`Installa o aggiorna kubectl`"

Procedura

1. Installa il plug-in del dispositivo Neuron Kubernetes usando Helm.

```
helm upgrade --install neuron-helm-chart oci://public.ecr.aws/neuron/neuron-helm-
chart \
    --set "npd.enabled=false"
```

2. Verifica che il plug-in del dispositivo Neuron sia in esecuzione. DaemonSet

```
kubectl get ds -n kube-system neuron-device-plugin
```

```
NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
neuron-device-plugin   1         1         1       1            1           <none>
       60s
```

3. Verifica che i tuoi nodi dispongano di dispositivi Neuron allocabili.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,NeuronDevice:.status.allocatable.aws\.amazon\.com/
neuron,NeuronCore:.status.allocatable.aws\.amazon\.com/neuroncore"
```

```
NAME                                          NeuronDevice   NeuronCore
ip-192-168-47-173.us-west-2.compute.internal   1              2
```

Verifica i dispositivi Neuron con un pod di prova

Puoi verificare che i dispositivi Neuron siano accessibili eseguendo lo `neuron-ls` strumento in un test Pod.

1. Crea un file denominato `neuron-ls.yaml` con i seguenti contenuti. Questo manifest avvia un contenitore Neuron Monitor in cui è installato lo `neuron-ls` strumento.

```
apiVersion: v1
kind: Pod
metadata:
  name: neuron-ls
spec:
  restartPolicy: Never
  containers:
  - name: neuron-container
    image: public.ecr.aws/g4h4h0b5/neuron-monitor:1.0.0
    command: ["/bin/sh"]
    args: ["-c", "neuron-ls"]
    resources:
      limits:
        aws.amazon.com/neuron: 1
      requests:
        aws.amazon.com/neuron: 1
  tolerations:
  - key: "aws.amazon.com/neuron"
    operator: "Exists"
    effect: "NoSchedule"
```

2. Applicare il file manifesto.

```
kubectl apply -f neuron-ls.yaml
```

3. Al termine dell'esecuzione del Pod, visualizza i suoi registri.

```
kubectl logs neuron-ls
```

Di seguito viene riportato un output di esempio.

```
instance-type: inf2.xlarge
instance-id: ...
+--------+--------+--------+---------+
| NEURON | NEURON | NEURON |   PCI   |
| DEVICE | CORES  | MEMORY |   BDF   |
+--------+--------+--------+---------+
| 0      | 2      | 32 GB  | 00:1f.0 |
+--------+--------+--------+---------+
```

> **Note**
>
> Quando si utilizza il plug-in per dispositivi Neuron, l'allocazione contigua dei dispositivi su istanze con più dispositivi Neuron (ad esempio`trn2.48xlarge`) richiede l'estensione dello scheduler Neuron Kubernetes. https://awsdocs-neuron.readthedocs-hosted.com/en/latest/containers/tutorials/k8s-neuron-scheduler.html Il driver Neuron DRA lo gestisce automaticamente tramite vincoli topologici.

Per ulteriori informazioni sull'utilizzo dei dispositivi Neuron con Amazon EKS, consulta la documentazione di Neuron per l'esecuzione su EKS.

## Gestisci le GPU NVIDIA su Amazon EKS

Le GPU NVIDIA sono ampiamente utilizzate per l'addestramento all'apprendimento automatico, l'inferenza e i carichi di lavoro di calcolo ad alte prestazioni. Amazon EKS supporta due meccanismi per la gestione dei dispositivi GPU NVIDIA nei cluster EKS: il driver NVIDIA DRA per GPU e il plug-in per dispositivi NVIDIA Kubernetes.

Consigliamo di utilizzare i driver DRA per le nuove implementazioni con Kubernetes versioni 1.34 e successive quando si utilizza il provisioning della capacità statica in Karpenter, gruppi di nodi gestiti da EKS o nodi autogestiti. https://karpenter.sh/docs/concepts/nodepools/#static-nodepool DRA non è

attualmente supportato con EKS Auto Mode. Il driver NVIDIA DRA consente un'allocazione flessibile delle GPU e la condivisione delle GPU tra contenitori.

Per la disponibilità delle funzionalità DRA in EKS, consulta le note di rilascio per le versioni di Kubernetes per EKS. Per ulteriori informazioni sull'utilizzo del driver e del plug-in del dispositivo NVIDIA DRA con EKS, vedere. the section called "DRA e plug-in per dispositivi"

## Condivisione della GPU (MIG time-slicing) &

### Multi-instance GPU (MIG)

Multi-Instance La GPU (MIG) è una funzionalità hardware delle GPU NVIDIA che partiziona una singola GPU fisica in più istanze isolate. Il numero massimo di istanze dipende dalla GPU. Le GPU per data center come A100, H100, H200 e B200 supportano fino a sette istanze, mentre le GPU e D ne supportano meno. Blackwell-based g7 g7e Ogni istanza dispone di memoria, unità di elaborazione e larghezza di banda di memoria dedicate, pertanto un carico di lavoro eseguito su un'istanza non può influire su un carico di lavoro su un'altra. A differenza del time-slicing, che condivide una GPU tramite il time-multiplexing del software senza isolamento, il MIG fornisce memoria a livello hardware e isolamento dei guasti tra i pod.

Il MIG è più adatto all'inferenza multi-tenant, ai carichi di lavoro che richiedono una qualità del servizio prevedibile e agli ambienti con requisiti di conformità per una tenancy isolata dall'hardware. È disponibile sulle GPU NVIDIA Ampere (A100), Hopper (H100 e H200) e Blackwell. Sì AWS, questi sono i tipi di istanza e i tipi di istanza e P-family . Blackwell-based g7 g7e

Per ulteriori informazioni e passaggi per l'utilizzo di MIG con GPU NVIDIA ed EKS, vedere. the section called "Multi-instance GPU (MIG)"

### Time-slicing

Time-slicing consente a più Pod di condividere una singola GPU fisica NVIDIA configurando il plug-in del dispositivo NVIDIA o il driver DRA in modo da pubblicizzare più di uno slot programmabile per GPU. Lo scheduler Kubernetes colloca più Pod sulla stessa GPU e lo scheduler CUDA della GPU multiplexa i carichi di lavoro nel tempo.

Time-slicing è la strategia più semplice. GPU-sharing Utilizza solo software, non richiede hardware speciale e funziona su tutti i tipi di istanze GPU NVIDIA. AWS Time-slicing non offre l'isolamento della memoria o del calcolo tra Pod che condividono una GPU. È ideale per carichi di lavoro con un basso utilizzo della GPU, come i servizi di inferenza che rimangono inattivi tra le richieste o gli ambienti di

sviluppo in cui più utenti condividono una GPU. Per i carichi di lavoro che richiedono memoria a livello hardware e isolamento di elaborazione, utilizza invece MIG.

Per ulteriori informazioni e passaggi per l'utilizzo del time-slicing con GPU NVIDIA ed EKS, consulta. the section called "Time-slicing"

### Argomenti

- the section called "DRA e plug-in per dispositivi"
- the section called "Multi-instance GPU (MIG)"
- the section called "Time-slicing"

## Usa il driver o il plug-in del dispositivo NVIDIA DRA su Amazon EKS

Amazon EKS supporta due meccanismi per la gestione dei dispositivi GPU NVIDIA nei tuoi cluster EKS: il driver NVIDIA DRA per GPU e il plug-in per dispositivi NVIDIA Kubernetes.

Consigliamo di utilizzare il driver NVIDIA DRA per le nuove implementazioni con Kubernetes versioni 1.34 e successive quando si utilizza il provisioning della capacità statica in Karpenter, gruppi di nodi gestiti da EKS o nodi autogestiti. https://karpenter.sh/docs/concepts/nodepools/#static-nodepool Il driver NVIDIA DRA non è attualmente supportato con EKS Auto Mode. Usa il plug-in del dispositivo NVIDIA con EKS Auto Mode o con Karpenter quando usi il provisioning dinamico della capacità. Il plug-in per dispositivi NVIDIA rimane supportato anche per i gruppi di nodi gestiti da EKS e i nodi autogestiti.

Se utilizzi funzionalità di condivisione GPU come GPU multiistanza (MIG) o time-slicing, ti consigliamo di utilizzare la configurazione statica con il driver DRA o il plug-in del dispositivo NVIDIA. Il MIG dinamico e il time-slicing sono in stato alfa nel driver NVIDIA DRA. Consulta le versioni dei driver NVIDIA DRA per gli ultimi aggiornamenti. GitHub

Driver NVIDIA DRA e plug-in per dispositivi NVIDIA

| Funzionalità | Driver NVIDIA DRA | Plugin per dispositivi NVIDIA |
|---|---|---|
| Versione minima di Kubernetes | 1.34 | Tutte le versioni di EKS-supported Kubernetes |
| EKS Compute | Karpenter (solo capacità statica), gruppi di nodi gestiti, nodi autogestiti | EKS Auto Mode, Karpenter, gruppi di nodi gestiti, nodi autogestiti |

| Funzionalità | Driver NVIDIA DRA | Plugin per dispositivi NVIDIA |
|---|---|---|
| EKS-optimized AMI | AL 2023 (NVIDIA), Bottlerocket | AL 2023 (NVIDIA), Bottlerocket |
| Pubblicità sul dispositivo | Attributi avanzati tramite `ResourceSlice` oggetti tra cui modello GPU, memoria, versione del driver e topologia | Numero intero di risorse estese `nvidia.com/gpu` |
| Condivisione GPU | (Alpha) MIG dinamico, MPS, time-slicing | (GA) MIG statico, MPS, time-slicing |
| ComputeDomains | Gestisce Multi-Node NVLink (MNNVL) tramite risorse per comunicazioni GPU multinodo sicure `ComputeDomain` | Non supportata |
| Attribute-based selezione | Filtrare le GPU per modello, memoria o altri attributi utilizzando le espressioni CEL | Non supportata |
| Topology-aware Allocazione EFA | DRA-native consapevolezza della topologia | Riconoscimento automatico della topologia (solo AMI AL2023) EKS-optimized |

Installa il driver NVIDIA DRA

Il driver NVIDIA DRA per GPU gestisce due tipi di risorse: GPU e. ComputeDomains Esegue due plugin DRA kubelet: e. `gpu-kubelet-plugin` `compute-domain-kubelet-plugin` Ciascuno può essere abilitato o disabilitato separatamente durante l'installazione. Questa guida si concentra sull'allocazione delle GPU. Per l'utilizzo ComputeDomains, vedere. the section called "Usa P6e-GB200 con EKS"

Prerequisiti

- Un cluster Amazon EKS che esegue Kubernetes versione 1.34 o successiva con capacità statica fornita da Karpenter, gruppi di nodi gestiti da EKS o gruppi di nodi autogestiti.

- Nodi con tipi di istanze GPU NVIDIA (ad esempio istanze or). `P G`
- Nodi con componenti a livello di host installati per le GPU NVIDIA. Quando si utilizzano le AMI NVIDIA EKS-optimized AL2023 o Bottlerocket, il driver NVIDIA a livello di host, il driver in modalità utente CUDA e il toolkit contenitore sono preinstallati.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurato per comunicare con il cluster, consulta per ulteriori informazioni. the section called "`Installa o aggiorna kubectl`"

Procedura

**Important**

Quando si utilizza il driver NVIDIA DRA per la gestione dei dispositivi GPU, non installarlo insieme al plug-in del dispositivo NVIDIA sullo stesso nodo. Ciò può causare una sottoscrizione silenziosa dei dispositivi sottostanti a più pod sullo stesso nodo.

**Disattiva il plug-in del dispositivo NVIDIA integrato su Bottlerocket**

Le varianti NVIDIA di EKS-optimized Bottlerocket includono il plug-in del dispositivo NVIDIA e lo abilitano per impostazione predefinita. Il driver DRA non può essere eseguito insieme al plug-in del dispositivo sullo stesso nodo. Prima di utilizzare il driver DRA, disabilitate il plug-in del dispositivo integrato nei nodi GPU Bottlerocket. Impostato su `false` nei `settings.kubelet-device-plugins.nvidia.enabled` dati utente del nodo Bottlerocket.

```
[settings.kubelet-device-plugins.nvidia]
enabled = false
```

L'`settings.kubelet-device-plugins.nvidia.enabled`impostazione è disponibile nella versione Bottlerocket 1.63.0 e successive. Nelle versioni precedenti, il plug-in integrato del dispositivo NVIDIA non può essere disabilitato. Per maggiori informazioni, vedi bottlerocket- os/bottlerocket pull request #4856 on. GitHub

1. Installa il driver NVIDIA DRA direttamente dal registro Kubernetes SIG OCI. Per trovare le versioni disponibili, consulta le versioni dei driver NVIDIA DRA su. https://github.com/kubernetes-sigs/dra-driver-nvidia-gpu/releases GitHub

```
helm install dra-driver-nvidia-gpu \
    oci://registry.k8s.io/dra-driver-nvidia/charts/dra-driver-nvidia-gpu \
    --version 0.4.1 \
    --create-namespace \
    --namespace nvidia \
    --set resources.computeDomains.enabled=false \
    --set gpuResourcesEnabledOverride=true
```

   Per le opzioni di configurazione avanzate, consultate i valori del grafico Helm del driver NVIDIA DRA sul sito web di Kubernetes SIG. Per visualizzare i valori disponibili per una versione specifica del grafico, esegui. `helm show values oci://registry.k8s.io/dra-driver-nvidia/charts/dra-driver-nvidia-gpu --version 0.4.1`

2. (Facoltativo) Per utilizzare il time-slicing tramite il driver DRA, aggiungete il `TimeSlicingSettings` feature gate al `helm install` comando nel passaggio precedente. Si tratta di una funzionalità alfa disattivata per impostazione predefinita. Per ulteriori informazioni, consulta the section called "Usa il time-slicing della GPU con il driver NVIDIA DRA".

```
--set featureGates.TimeSlicingSettings=true
```

3. (Facoltativo) Per utilizzare il MIG dinamico tramite il driver DRA, aggiungete il `DynamicMIG` feature gate al `helm install` comando nel passaggio precedente. Questa è una funzionalità alfa disattivata per impostazione predefinita. Non è possibile combinare il `DynamicMIG` feature gate con i feature gate `PassthroughSupportNVMLDeviceHealthCheck`,, o `MPSSupport` feature. Per ulteriori informazioni, consulta the section called "Usa MIG con il driver NVIDIA DRA".

```
--set featureGates.DynamicMIG=true
```

4. Verificare che i driver pod DRA siano in funzione.

```
kubectl get pods -n nvidia
```

5. Verificate che gli `DeviceClass` oggetti siano stati creati.

```
kubectl get deviceclass
```

```
NAME            AGE
gpu.nvidia.com  60s
```

6. Verifica che `ResourceSlice` gli oggetti siano pubblicati per i tuoi nodi GPU.

```
kubectl get resourceslice
```

Per richiedere le GPU NVIDIA utilizzando il driver DRA, creane un riferimento `gpu.nvidia.com` `DeviceClass` e `ResourceClaimTemplate` inseriscilo nelle specifiche del tuo Pod. L'esempio seguente richiede una singola GPU. Consulta the section called "Topology-aware EFA e GPU/ Neuron allocazione dei dispositivi" la procedura per allocare le GPU NVIDIA con interfacce EFA allineate alla topologia.

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: single-gpu
spec:
  spec:
    devices:
      requests:
      - name: gpu
        exactly:
          deviceClassName: gpu.nvidia.com
          count: 1
---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-workload
spec:
  containers:
  - name: gpu-demo
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    command: ["/bin/sh", "-c"]
    args: ["nvidia-smi && tail -f /dev/null"]
    resources:
      claims:
      - name: gpu
  resourceClaims:
  - name: gpu
```

```
    resourceClaimTemplateName: single-gpu
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
```

Installa il plug-in del dispositivo NVIDIA Kubernetes

Il plug-in per dispositivi NVIDIA Kubernetes pubblicizza le GPU NVIDIA come risorse estese. `nvidia.com/gpu` Richiedete le GPU in container, richieste e limiti di risorse.

Prerequisiti

- Un cluster Amazon EKS.
- Nodi con tipi di istanze GPU NVIDIA (ad esempio `G` istanze `P` or).
- Nodi con tipi di istanze GPU NVIDIA che utilizzano l'AMI NVIDIA AL2023. EKS-optimized Le AMI EKS-optimized Bottlerocket includono il plug-in del dispositivo NVIDIA. Non è necessario installarlo separatamente.
- Nodi con componenti a livello di host installati per le GPU NVIDIA. Quando si utilizzano le AMI NVIDIA EKS-optimized AL2023 o Bottlerocket, il driver NVIDIA a livello di host, il driver in modalità utente CUDA e il toolkit contenitore sono preinstallati.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurato per comunicare con il cluster, consulta per ulteriori informazioni. the section called "`Installa o aggiorna kubectl`"

Procedura

1. Aggiungi il repository grafico Helm del plug-in per dispositivi NVIDIA.

```
helm repo add nvdp https://nvidia.github.io/k8s-device-plugin
```

2. Aggiorna il tuo repository Helm locale.

```
helm repo update
```

3. Installa il plug-in del dispositivo NVIDIA Kubernetes.

```
helm install nvdp nvdp/nvidia-device-plugin \
    --create-namespace \
    --namespace nvidia
```

Opzionale: abilita GPU Feature Discovery (GFD)

Il plug-in del dispositivo pubblicizza risorse `nvidia.com/gpu` estese e pianifica autonomamente i GPU Pod. Sulla NVIDIA AMI EKS-optimized AL2023, l'etichetta del `nvidia.com/gpu.present=true` nodo è già applicata all'avvio`nodeadm`, quindi GPU Feature Discovery (GFD) non è richiesto per la pianificazione di base della GPU.
Abilita GFD con `--set gfd.enabled=true` se desideri che il nodo sia etichettato con attributi GPU dettagliati, come,, etichette del profilo MIG e versioni. `nvidia.com/gpu.product` `nvidia.com/gpu.memory` `nvidia.com/gpu.count` driver/CUDA
Con queste etichette, puoi scegliere come target specifici tipi di GPU con affinità di nodo (ad esempio, programmando un carico di lavoro solo sulle GPU A10G `nodeSelector` o su un particolare profilo MIG). Anche le configurazioni di condivisione della GPU come time-slicing e MIG utilizzano queste etichette. Se non hai bisogno della selezione dei nodi basata sugli attributi o della condivisione della GPU, puoi omettere il flag.

```
helm install nvdp nvdp/nvidia-device-plugin \
    --create-namespace \
    --namespace nvidia \
    --set gfd.enabled=true
```

Abilita GDRCopy se i tuoi carichi di lavoro lo utilizzano

`k8s-device-plugin`La versione da v0.19.0 a v0.19.2 ha abilitato le funzionalità GDRCopy e MOFED per impostazione predefinita. Questa abilitazione predefinita è stata ripristinata nella `k8s-device-plugin` versione 0.19.3. Come risultato del ripristino, GDRCopy (`gdrdrv`) non può più essere abilitato per contenitore con la variabile di `NVIDIA_GDRCOPY=enabled` ambiente in una specifica Pod, tale variabile viene ora ignorata.
Se i tuoi carichi di lavoro utilizzano GDRCopy (copia GPUDirect RDMA), devi abilitarlo sul plug-in del dispositivo al momento dell'installazione impostando: `gdrcopyEnabled=true`

```
helm upgrade --install nvdp nvdp/nvidia-device-plugin \
    --namespace nvidia \
    --create-namespace \
    --set gdrcopyEnabled=true
```

(Aggiungetelo `--set gfd.enabled=true` anche se volete aggiungere anche le etichette GPU Feature Discovery, come descritto in precedenza).
Se gestisci il plug-in del dispositivo NVIDIA tramite NVIDIA GPU Operator attivo GitHub, l'operatore imposta dinamicamente `GDRCOPY_ENABLED=true` quando il modulo del `gdrdrv` kernel viene caricato sul nodo.
Per ulteriori informazioni, consulta il numero #1692 di NVIDIA k8s-device-plugin su. GitHub

> **Note**
>
> Puoi anche installare e gestire il plug-in del dispositivo NVIDIA Kubernetes utilizzando NVIDIA GPU Operator on GitHub, che automatizza la gestione di tutti i componenti software NVIDIA necessari per il provisioning delle GPU.

4. Verifica che il plug-in del dispositivo NVIDIA DaemonSet sia in esecuzione.

```
kubectl get ds -n nvidia nvdp-nvidia-device-plugin
```

```
NAME                        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
nvdp-nvidia-device-plugin   2         2         2       2            2
 <none>          60s
```

5. Verifica che i tuoi nodi dispongano di GPU allocabili.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

Di seguito viene riportato un output di esempio.

```
NAME                                           GPU
```

```
ip-192-168-11-225.us-west-2.compute.internal   1
ip-192-168-24-96.us-west-2.compute.internal    1
```

### Richiedi le GPU NVIDIA in un pod

Per richiedere le GPU NVIDIA utilizzando il plug-in del dispositivo, specifica la `nvidia.com/gpu` risorsa nel contenitore (richieste e limiti).

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: gpu-demo
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    command: ["/bin/sh", "-c"]
    args: ["nvidia-smi && tail -f /dev/null"]
    resources:
      limits:
        nvidia.com/gpu: 1
      requests:
        nvidia.com/gpu: 1
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Equal"
    value: "true"
    effect: "NoSchedule"
```

Per eseguire questo test, applica il manifest e visualizza i log:

```
kubectl apply -f nvidia-smi.yaml
kubectl logs nvidia-smi
```

Di seguito viene riportato un output di esempio.

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI XXX.XXX.XX            Driver Version: XXX.XXX.XX     CUDA Version: XX.X
   |
```

```
|-----------------------------------------+------------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
 ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute
 M. |
|                                         |                        |               MIG
 M. |
|=========================================+========================
+======================|
|   0  NVIDIA L4                      On  |   00000000:31:00.0 Off |
 0 |
| N/A   27C    P8             11W /   72W |       0MiB /  23034MiB |      0%
 Default |
|                                         |                        |
 N/A |
+-----------------------------------------+------------------------
+----------------------+

+---------------------------------------------------------------------------------------
+
| Processes:
   |
|  GPU   GI   CI              PID   Type   Process name                        GPU
 Memory |
|        ID   ID                                                               Usage
   |
|
=======================================================================================|
|  No running processes found
   |
+---------------------------------------------------------------------------------------
+
```

## Usa GPU multiistanza (MIG) con GPU NVIDIA su Amazon EKS

Multi-Instance La GPU (MIG) è una funzionalità hardware delle GPU NVIDIA che partiziona una singola GPU fisica in più istanze isolate. Il numero massimo di istanze partizionate dipende dalla GPU. Ogni istanza dispone di memoria, unità di calcolo e larghezza di banda di memoria dedicate, pertanto un carico di lavoro eseguito su un'istanza non può influire su un carico di lavoro su un'altra. A differenza del time-slicing, che condivide una GPU tramite il time-multiplexing del software senza isolamento, il MIG fornisce memoria a livello hardware e isolamento dei guasti tra i pod.

MIG è più adatto per l'inferenza multi-tenant e per i carichi di lavoro che richiedono una qualità del servizio prevedibile. È disponibile sulle GPU NVIDIA Ampere (A100), Hopper (H100 e H200) e Blackwell. Sì AWS, questi sono i tipi di istanza e i tipi di istanza e P-family . Blackwell-based `g7` `g7e` Per l'elenco completo, consulta the section called "MIG-capable tipi di istanza".

Il MIG è adatto quando:

- È necessario isolare la memoria, in modo che un carico di lavoro non possa consumare la memoria GPU richiesta da un altro carico di lavoro.
- Si esegue un'inferenza multi-tenant in cui i tenant condividono l'hardware della GPU ma richiedono una qualità del servizio per tenant.
- Esegui già il training su istanze A100, H100, H200 o Blackwell e desideri riutilizzare tali GPU per carichi di lavoro di inferenza più piccoli quando l'addestramento è inattivo.

Prendi in considerazione un approccio diverso quando:

- I nodi utilizzano un tipo di istanza GPU che non supporta MIG, ad esempio le famiglie`g5`,`g6`, o`g6e`. Usa invece il time-slicing.
- Non è necessario l'isolamento della memoria e si desidera la configurazione più semplice. Usa invece il time-slicing.
- È necessario modificare frequentemente le partizioni della GPU senza interruzioni. La modifica della modalità MIG o del layout delle partizioni richiede un ripristino della GPU, che l'operatore GPU esegue riavviando il nodo.
- Si esegue un training su più GPU che dipende dalla comunicazione collettiva o peer-to-peer tra le GPU. MIG non supporta NCCL o P2P tra GPU.

Considerazioni

Esamina le seguenti considerazioni prima di utilizzare MIG in produzione.

Considerazioni generali

- La modifica della configurazione MIG richiede un ripristino della GPU.  L'attivazione o la disattivazione della modalità MIG o la modifica del layout della partizione richiede un ripristino della GPU, quindi non può essere modificato sul posto. Ad esempio, il MIG Manager di NVIDIA GPU Operator applica una modifica alla configurazione arrestando i GPU Pod sul nodo e riavviando il nodo quando è necessario un riavvio per cambiare la modalità MIG.

- L'elaborazione non è strettamente proporzionale alla dimensione dell'istanza. Un'1gistanza non fornisce una quota proporzionale del throughput dell'intera GPU per ogni carico di lavoro, poiché la larghezza di banda della memoria e il comportamento della cache differiscono a seconda dei profili. Esegui il benchmark del tuo carico di lavoro sul profilo che intendi utilizzare prima di dimensionare le partizioni.
- Time-slicing non ha effetto sulle istanze MIG. Un'istanza MIG è già isolata dall'hardware e non può essere ulteriormente condivisa tramite time-slicing. La richiesta della strategia di `TimeSlicing` condivisione su un dispositivo MIG non modifica il comportamento dell'hardware. Per condividere una singola istanza MIG tra container, usa invece NVIDIA Multi-Process Service (MPS).
- Comunicazione tra GPU limitata. Quando il MIG è abilitato, le istanze MIG su GPU diverse non possono utilizzare la comunicazione GPU-to-GPU peer-to-peer (P2P) e NCCL non funziona con MIG. Multi-GPU i carichi di lavoro che dipendono dalla comunicazione collettiva o dal P2P tra le GPU, come l'addestramento di più GPU in parallelo con tensori, richiedono invece GPU intere. Per i dettagli, consulta le Considerazioni sulle applicazioni nella Guida per l'utente di NVIDIA MIG sul sito Web di NVIDIA.

Considerazioni sui plug-in dei dispositivi NVIDIA

- Le richieste di risorse dei pod devono corrispondere alla strategia. Con l'unica strategia, Pods request`nvidia.com/gpu`. Con la strategia mista, i Pods richiedono la risorsa specifica del profilo, ad esempio. `nvidia.com/mig-1g.10gb` Un Pod che richiede un profilo che il nodo non pubblicizza rimane nello stato. `Pending` Conferma le risorse pubblicizzate con. `kubectl describe node <node-name>`
- Plugin per dispositivi standalone su AL2023. Se installi il plug-in del dispositivo NVIDIA separatamente, ad esempio come parte della configurazione del cluster, escludilo dai tuoi nodi MIG in modo che non entri in conflitto con il plug-in del dispositivo gestito dall'operatore GPU. Aggiungi una regola di affinità dei nodi al plug-in del dispositivo autonomo che escluda i nodi con l'etichetta. `nvidia.com/mig.config`
- Nessun supporto per la modalità EKS Auto. EKS Auto Mode gestisce il plug-in del dispositivo NVIDIA e non ne espone la configurazione (vederethe section called "Implementare carico di lavoro accelerato"). Non è possibile abilitare MIG sui nodi EKS Auto Mode. Configura MIG su nodi Karpenter autogestiti o su un gruppo di nodi gestiti, in cui controlli le impostazioni dell'AMI e del plug-in del dispositivo.

Considerazioni sui driver NVIDIA DRA

- Il MIG statico richiede istanze precreate.  Con il MIG statico, il driver DRA alloca le istanze MIG esistenti ma non abilita la modalità MIG né partiziona le GPU. È necessario abilitare la modalità MIG e creare prima le istanze, ad esempio con MIG Manager in NVIDIA GPU Operator o. `nvidia-smi` Per ulteriori informazioni, consulta the section called “Usa MIG con il driver NVIDIA DRA”.
- Dynamic MIG è una funzionalità alpha.  Con il MIG dinamico, il driver crea e distrugge partizioni MIG su richiesta in risposta alle richieste di carico di lavoro. Richiede il `DynamicMIG` feature gate, che è disabilitato per impostazione predefinita. Per ulteriori informazioni, consulta the section called “Usa MIG con il driver NVIDIA DRA”.
- Disabilita il plug-in del dispositivo integrato su Bottlerocket.  Il driver DRA non può essere eseguito insieme al plug-in del dispositivo NVIDIA sullo stesso nodo. Su Bottlerocket, disabilita il plug-in del dispositivo integrato, che richiede Bottlerocket versione 1.63.0 o successiva. Per ulteriori informazioni, consulta the section called “Installa il driver NVIDIA DRA”.
- Supporto informatico.  Il driver NVIDIA DRA è supportato con il provisioning statico della capacità in Karpenter, gruppi di nodi gestiti EKS o nodi autogestiti e non è supportato con EKS Auto Mode. Per ulteriori informazioni, consulta la documentazione statica di Karpenter sul sito Web di  Karpenter. NodePool

MIG-capable tipi di istanza

Attivata AWS, i seguenti tipi di istanza forniscono MIG-capable GPU.

| Tipo di istanza | GPU | Memoria GPU |
|---|---|---|
| `p4d.24xlarge` | 8 x NVIDIA A100 da 40 GB | 320 GB |
| `p4de.24xlarge` | 8 x NVIDIA A100 da 80 GB | 640 GB |
| `p5.48xlarge` | 8 x NVIDIA H100 da 80 GB | 640 GB |
| `p5e.48xlarge` | 8 x NVIDIA H200 | 128 GB |
| `p5en.48xlarge` | 8 x NVIDIA H200 | 128 GB |
| `p6-b200.48xlarge` | 8 x NVIDIA Blackwell B200 | 1432 GB |

| Tipo di istanza | GPU | Memoria GPU |
|---|---|---|
| p6-b300.48xlarge | 8 x NVIDIA Blackwell Ultra B300 | 2144 GB |
| g7.48xlarge | 8 x NVIDIA RTX PRO 4500 Blackwell Server Edition | 256 GB |
| g7e.48xlarge | 8 x NVIDIA RTX PRO 6000 Blackwell Server Edition | 768 GB |

MIG non è disponibile nelle famiglieg5,g6, og6e. Per chi utilizza p6e-gb200 UltraServers la MIG-capable GPU NVIDIA GB200, vedere. the section called "Usa P6e-GB200 con EKS"

> **Note**
>
> Il tipo di g7 istanza richiede la versione 595 del driver NVIDIA o successiva. Le AMI EKS-optimized accelerate attualmente includono il driver NVIDIA versione 580, quindi per utilizzare MIG su è g7 necessario creare un'AMI personalizzata con la versione del driver 595. Per ulteriori informazioni, consulta the section called "Crea un'AMI EKS-optimized Amazon Linux personalizzata".

Le istanze MIG sono descritte da profili che utilizzano il modello di denominazione<slices>g.<memory>gb, dove <slices> è il numero di slice di calcolo e la memoria dell'istanza in gigabyte. <memory> Ad esempio, il 3g.40gb profilo fornisce tre delle sette slice di calcolo e 40 GB di memoria. I profili supportati da ciascuna GPU vengono fissati dall'hardware. Per l'elenco completo, consulta la Guida per l'utente della Multi-Instance GPU NVIDIA sul sito Web di NVIDIA.

Profili MIG per tipo di istanza

I profili MIG disponibili su un nodo dipendono dalla GPU per il tipo di istanza. Le sezioni seguenti elencano i profili per ogni tipo di istanza MIG-capable Amazon EC2. Per ogni profilo, il numero massimo di istanze è il numero massimo di istanze di quel profilo che puoi creare su una singola GPU e Memoria per istanza è la memoria GPU allocata a ciascuna di esse.

p4d.24xlarge — NVIDIA A100 40 GB

| Profilo | Fette di calcolo | Memoria per istanza | Numero massimo di istanze |
|---|---|---|---|
| `1g.5gb` | 1 di 7 | 5 GB | 7 |
| `1g.10gb` | 1 di 7 | 10 GB | 4 |
| `2g.10gb` | 2 di 7 | 10 GB | 3 |
| `3g.20gb` | 3 di 7 | 20 GB | 2 |
| `4g.20gb` | 4 di 7 | 20 GB | 1 |
| `7g.40gb` | 7 di 7 | 40 GB | 1 |

p4de.24xlarge — NVIDIA A100 80 GB

| Profilo | Sezioni di calcolo | Memoria per istanza | Numero massimo di istanze |
|---|---|---|---|
| `1g.10gb` | 1 di 7 | 10 GB | 7 |
| `1g.20gb` | 1 di 7 | 20 GB | 4 |
| `2g.20gb` | 2 di 7 | 20 GB | 3 |
| `3g.40gb` | 3 di 7 | 40 GB | 2 |
| `4g.40gb` | 4 di 7 | 40 GB | 1 |
| `7g.80gb` | 7 di 7 | 80 GB | 1 |

**`p5.48xlarge`** — NVIDIA H100 80 GB

| Profilo | Fette di calcolo | Memoria per istanza | Numero massimo di istanze |
|---|---|---|---|
| `1g.10gb` | 1 di 7 | 10 GB | 7 |
| `1g.20gb` | 1 di 7 | 20 GB | 4 |
| `2g.20gb` | 2 di 7 | 20 GB | 3 |
| `3g.40gb` | 3 di 7 | 40 GB | 2 |
| `4g.40gb` | 4 di 7 | 40 GB | 1 |
| `7g.80gb` | 7 di 7 | 80 GB | 1 |

**`p5e.48xlarge`** e  p5en.48xlarge — NVIDIA H200 141 GB

| Profilo | Sezioni di calcolo | Memoria per istanza | Numero massimo di istanze |
|---|---|---|---|
| `1g.18gb` | 1 di 7 | 18 GB | 7 |
| `1g.35gb` | 1 di 7 | 35 GB | 4 |
| `2g.35gb` | 2 di 7 | 35 GB | 3 |
| `3g.71gb` | 3 di 7 | 71 GB | 2 |
| `4g.71gb` | 4 di 7 | 71 GB | 1 |
| `7g.141gb` | 7 di 7 | 141 GB | 1 |

p6-b200.48xlarge — NVIDIA Blackwell B200 da 180 GB

| Profilo | Fette di calcolo | Memoria per istanza | Numero massimo di istanze |
|---|---|---|---|
| `1g.23gb` | 1 di 7 | 23 GB | 7 |
| `1g.45gb` | 1 di 7 | 45 GB | 4 |
| `2g.45gb` | 2 di 7 | 45 GB | 3 |
| `3g.90gb` | 3 di 7 | 90 GB | 2 |
| `4g.90gb` | 4 di 7 | 90 GB | 1 |
| `7g.180gb` | 7 di 7 | 180 GB | 1 |

p6-b300.48xlarge — NVIDIA Blackwell Ultra B300

`p6-b300.48xlarge`Utilizza l'HGX B300, che supporta il partizionamento di ciascuna GPU in 7 istanze da 32 GB, 4 da 67 GB, 2 da 135 GB o 1 da 270 GB. Queste dimensioni sono preliminari e potrebbero cambiare. Per i dettagli del profilo, consulta i profili MIG supportati da NVIDIA sul sito Web di NVIDIA.

**`g7.48xlarge`** — NVIDIA RTX PRO 4500 Blackwell Server Edition da 32 GB

| Profilo | Sezioni di calcolo | Memoria per istanza | Numero massimo di istanze |
|---|---|---|---|
| `1g.16gb` | 1 di 2 | 16 GB | 2 |
| `2g.32gb` | 2 di 2 | 32 GB | 1 |

L'RTX PRO 4500 Blackwell supporta anche varianti di profilo abilitate alla grafica (`+gfx`) e media-engine (,). `+me` `.all` `-me` Per l'elenco completo, consulta i profili MIG supportati da NVIDIA sul sito Web di NVIDIA.

**`g7e.48xlarge`** — NVIDIA RTX PRO 6000 Blackwell Server Edition 96 GB

| Profilo | Sezioni di calcolo | Memoria per istanza | Numero massimo di istanze |
|---|---|---|---|
| `1g.24gb` | 1 di 4 | 24 GB | 4 |
| `2g.48gb` | 2 di 4 | 48 GB | 2 |
| `4g.96gb` | 4 di 4 | 96 GB | 1 |

L'RTX PRO 6000 Blackwell Server Edition supporta anche varianti di profilo abilitate alla grafica (`+gfx`) e media-engine (,). `+me.all -me` Per l'elenco completo, consulta i profili MIG supportati da NVIDIA sul sito Web di NVIDIA.

Strategie MIG

Il driver NVIDIA DRA e il plug-in del dispositivo NVIDIA espongono le istanze MIG a Kubernetes in diversi modi. Il plug-in del dispositivo utilizza un'impostazione della strategia MIG a livello di nodo, mentre il driver DRA non ha un'impostazione equivalente perché seleziona le istanze in base ai relativi attributi. Comprendere questa differenza è fondamentale per scegliere tra i due modelli.

Driver NVIDIA DRA

Il driver NVIDIA DRA non utilizza il concetto di strategia singola o mista e non esiste un'impostazione equivalente da configurare. Invece di pubblicizzare le istanze MIG come risorse conteggiate, il driver pubblica ogni istanza come dispositivo `mig.nvidia.com` `DeviceClass` con attributi come il suo. `profile` I pod selezionano un'istanza facendo corrispondere questi attributi ai selettori CEL (Common Expression Language) in un `ResourceClaim` o`ResourceClaimTemplate`, come mostrato in. the section called “Usa MIG con il driver NVIDIA DRA”

Poiché la selezione è per istanza, i nodi a profilo misto funzionano senza un cambio di modalità strategica. Una singola GPU può essere suddivisa in diversi profili e ogni claim seleziona il profilo di cui ha bisogno. La scelta che conta per il driver DRA non è la strategia MIG singola o mista, ma la strategia MIG statica o dinamica, che determina se le istanze MIG sono precreate o il driver le crea su richiesta. Per ulteriori informazioni, consulta the section called “Usa MIG con il driver NVIDIA DRA”.

### Plugin per dispositivi NVIDIA

Il plug-in per dispositivi NVIDIA pubblicizza le istanze MIG su Kubernetes utilizzando una delle due strategie. Poiché il plug-in del dispositivo espone le istanze MIG come risorse estese a livello di nodo, che contengono solo un conteggio di numeri interi e nessun attributo per istanza, la strategia determina il nome di tali risorse.

- Strategia singola: ogni GPU su un nodo utilizza lo stesso profilo MIG. Il plug-in del dispositivo pubblicizza ogni istanza come `nvidia.com/gpu` risorsa e i Pods richiedono `nvidia.com/gpu: 1` come farebbero per una GPU dedicata. I manifesti esistenti non vengono modificati. Sia Bottlerocket che AL2023 supportano la strategia unica.
- Strategia mista: le GPU sullo stesso nodo possono utilizzare diversi profili MIG. Il plug-in del dispositivo pubblicizza ogni profilo come una risorsa distinta, ad esempio `nvidia.com/mig-1g.10gb` or`nvidia.com/mig-3g.40gb`, e i Pods richiedono il profilo specifico di cui hanno bisogno. Non è possibile utilizzare una strategia mista con il plug-in per dispositivi NVIDIA integrato di Bottlerocket. Per ulteriori informazioni, consulta il numero #4483 di GitHub Bottlerocket su. GitHub

### Usa MIG con il driver NVIDIA DRA

Quando si allocano le istanze MIG con il driver NVIDIA DRA, i Pod richiedono un'istanza MIG tramite un `ResourceClaim` or anziché la risorsa estesa del plug-in del dispositivo. `ResourceClaimTemplate` `nvidia.com/mig-<profile>`

Poiché il driver DRA descrive le istanze in base ai relativi attributi anziché come risorse conteggiate, non utilizza la strategia singola o mista richiesta dal plug-in del dispositivo (vedere). the section called "Strategie MIG" Il driver espone ogni istanza MIG come dispositivo `mig.nvidia.com` `DeviceClass` con un `gpu.nvidia.com/type` attributo of `mig` e pubblicizza attributi per istanza come `profile` (ad esempio`1g.5gb`) e della GPU fisica. `parentUUID` È possibile abbinare questi attributi ai selettori CEL (Common Expression Language) per richiedere un profilo specifico o per mantenere più istanze sulla stessa GPU.

Il driver DRA alloca le istanze MIG in una delle due modalità seguenti:

- MIG statico: si abilita la modalità MIG e si creano le istanze MIG sul nodo prima dell'avvio del driver, ad esempio con il MIG Manager in NVIDIA GPU Operator, come descritto in. the section called "Usa MIG sui nodi AL2023 con il plug-in del dispositivo NVIDIA" Il driver rileva le istanze esistenti e le assegna ai Pods ma non modifica la configurazione MIG del nodo. Le istanze

aggiunte dopo l'avvio del driver non vengono scoperte fino al riavvio del plug-in GPU kubelet. Il MIG statico è l'impostazione predefinita e non richiede alcun feature gate.

- MIG dinamico: il driver crea e distrugge partizioni MIG su richiesta in risposta alle richieste di carico di lavoro, in modo da non partizionare le GPU in anticipo. Il MIG dinamico è una funzionalità alfa disattivata per impostazione predefinita. Si richiede un profilo con gli stessi `ResourceClaimTemplate` selettori mostrati nelle sezioni seguenti e il driver partiziona una GPU per soddisfare la richiesta.

### Considerazioni

- Il MIG dinamico sostituisce il rilevamento statico su un nodo. Il driver gestisce tutte le partizioni e distrugge tutte le partizioni MIG che non ha creato all'avvio del plug-in GPU kubelet. Non attivate il MIG dinamico sui nodi con partizioni predefinite che desiderate conservare e non eseguite `mig-parted` né `nvidia-smi mig` mentre il plugin è in esecuzione, perché le modifiche manuali possono entrare in conflitto con lo stato della partizione del driver e causare il fallimento della preparazione o della pulizia del Pod.
- Il MIG dinamico è in stato alfa e richiede l'attivazione di un feature gate durante l'installazione del driver NVIDIA DRA, vedi le istruzioni. the section called "Installa il driver NVIDIA DRA"
- Le architetture Hopper (H100 e H200) e successive abilitano la modalità MIG su richiesta. Le generazioni precedenti non possono abilitare la modalità MIG on demand, incluse le GPU Ampere (A100).
- Il MIG dinamico dipende dalla funzionalità dei dispositivi partizionabili di Kubernetes (attiva GitHub), abilitata per impostazione predefinita KEP-4815 nella versione 1.36 e successive di Kubernetes. Nelle versioni precedenti questa funzionalità non è abilitata per impostazione predefinita, quindi lo scheduler non può allocare dispositivi MIG creati dinamicamente.

### Prerequisiti

- Un cluster Amazon EKS che esegue Kubernetes versione 1.34 o successiva con capacità statica fornita da Karpenter, gruppi di nodi gestiti da EKS o gruppi di nodi autogestiti.
- MIG-capable P-family nodi con modalità MIG abilitata e GPU partizionate in istanze MIG. Per il MIG statico, consulta MIG Manager nella sezione NVIDIA GPU Operator sul sito Web di NVIDIA.
- Il driver NVIDIA DRA è stato installato come descritto inthe section called "Installa il driver NVIDIA DRA", opzionalmente con Dynamic MIG abilitato se non si utilizza il partizionamento MIG statico.

Procedura

I seguenti esempi possono essere utilizzati con MIG statico o dinamico e il driver NVIDIA DRA.

1. Crea un file `ResourceClaimTemplate` che richieda un'istanza MIG da e un Pod che `mig.nvidia.com` `DeviceClass` vi faccia riferimento. Questo esempio richiede qualsiasi istanza MIG disponibile senza vincolare il profilo.

```
cat <<EOF | kubectl apply -f -
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: mig-profile-any
spec:
  spec:
    devices:
      requests:
      - name: mig
        exactly:
          deviceClassName: mig.nvidia.com
          count: 1
---
apiVersion: v1
kind: Pod
metadata:
  name: mig-dra-pod
spec:
  tolerations:
    - key: nvidia.com/gpu
      operator: Exists
      effect: NoSchedule
  containers:
    - name: cuda
      image: nvidia/cuda:12.6.0-base-ubuntu22.04
      command: ["nvidia-smi", "-L"]
      resources:
        claims:
        - name: mig
  resourceClaims:
    - name: mig
      resourceClaimTemplateName: mig-profile-any
  restartPolicy: OnFailure
EOF
```

2. Verifica che al Pod sia stata assegnata una singola istanza MIG.

```
kubectl logs mig-dra-pod
```

Di seguito viene riportato un output di esempio. Il Pod vede un'istanza MIG dalla GPU partizionata.

```
GPU 0: NVIDIA A100-SXM4-40GB (UUID: GPU-edd63844-8488-f76a-f6e0-1027a7319a88)
  MIG 2g.10gb     Device  0: (UUID: MIG-a5ad493e-e7e8-5675-9381-1d0f5311a456)
```

Richiedi un profilo MIG specifico

Per richiedere un profilo specifico anziché qualsiasi istanza disponibile, aggiungi un selettore CEL che corrisponda all'`profile`attributo. Quanto segue `ResourceClaimTemplate` richiede un'`1g.5gb`istanza e il Pod vi fa riferimento.

```
cat <<EOF | kubectl apply -f -
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: mig-profile-1g.5gb
spec:
  spec:
    devices:
      requests:
      - name: mig
        exactly:
          deviceClassName: mig.nvidia.com
          selectors:
          - cel:
              expression: "device.attributes['gpu.nvidia.com'].profile == '1g.5gb'"
---
apiVersion: v1
kind: Pod
metadata:
  name: mig-profile-pod
spec:
  tolerations:
    - key: nvidia.com/gpu
      operator: Exists
      effect: NoSchedule
  containers:
```

```
    - name: cuda
      image: nvidia/cuda:12.6.0-base-ubuntu22.04
      command: ["nvidia-smi", "-L"]
      resources:
        claims:
        - name: mig
  resourceClaims:
    - name: mig
      resourceClaimTemplateName: mig-profile-1g.5gb
  restartPolicy: OnFailure
EOF
```

Richiedi più istanze MIG dalla stessa GPU

Per garantire che più istanze MIG in una singola richiesta provengano dalla stessa GPU fisica, aggiungi un blocco con. `constraints` `matchAttribute: "gpu.nvidia.com/parentUUID"` Quanto segue `ResourceClaimTemplate` richiede un'`1g.5gb`istanza e un'`2g.10gb`istanza dalla stessa GPU e il Pod fa riferimento al claim. Poiché il contenitore fa riferimento al claim senza nominare una richiesta specifica, riceve entrambe le istanze MIG.

```
cat <<EOF | kubectl apply -f -
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: multi-mig
spec:
  spec:
    devices:
      requests:
      - name: mig-small
        exactly:
          deviceClassName: mig.nvidia.com
          selectors:
          - cel:
              expression: "device.attributes['gpu.nvidia.com'].profile == '1g.5gb'"
      - name: mig-medium
        exactly:
          deviceClassName: mig.nvidia.com
          selectors:
          - cel:
              expression: "device.attributes['gpu.nvidia.com'].profile == '2g.10gb'"
      constraints:
      - requests: []
```

```
        matchAttribute: "gpu.nvidia.com/parentUUID"
---
apiVersion: v1
kind: Pod
metadata:
  name: multi-mig-pod
spec:
  tolerations:
    - key: nvidia.com/gpu
      operator: Exists
      effect: NoSchedule
  containers:
    - name: cuda
      image: nvidia/cuda:12.6.0-base-ubuntu22.04
      command: ["nvidia-smi", "-L"]
      resources:
        claims:
        - name: mig
  resourceClaims:
    - name: mig
      resourceClaimTemplateName: multi-mig
  restartPolicy: OnFailure
EOF
```

Usa MIG sui nodi Bottlerocket con il plug-in del dispositivo NVIDIA

Su Bottlerocket, l'AMI accelerata include il EKS-optimized plug-in del dispositivo NVIDIA. È possibile abilitare MIG con la singola strategia tramite le impostazioni. `settings.kubelet-device-plugins.nvidia` Bottlerocket supporta MIG nella versione 1.34.0 e successive.

Prerequisiti

- Un cluster Amazon EKS. La procedura seguente esegue il provisioning MIG-capable P-family dei nodi con l'AMI NVIDIA EKS-optimized Bottlerocket, versione 1.34.0 o successiva.
- Karpenter è stato installato e configurato nel cluster, poiché la procedura seguente utilizza un Karpenter per fornire le impostazioni MIG nei dati utente del nodo `EC2NodeClass` Bottlerocket. Per ulteriori informazioni, consulta la Guida introduttiva a Karpenter sul sito Web di Karpenter.
- `kubectl`configurato per comunicare con il tuo cluster, vedi the section called "`Installa o aggiorna kubectl`" per ulteriori informazioni.

Procedura

Aggiungi l'impostazione di partizionamento MIG ai dati utente di Bottlerocket per i tuoi nodi GPU. Il modo in cui fornisci i dati utente dipende dal modo in cui esegui il provisioning dei nodi. L'esempio seguente mostra un Karpenter `EC2NodeClass` per i `p4d.24xlarge` nodi.

```
cat <<EOF | kubectl apply -f -
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-bottlerocket-mig
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - alias: bottlerocket@latest
  role: eksctl-KarpenterNodeRole-<cluster-name>
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: <cluster-name>
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: <cluster-name>
  userData: |
    [settings.kubelet-device-plugins.nvidia]
    device-partitioning-strategy = "mig"

    [settings.kubelet-device-plugins.nvidia.mig.profile]
    "a100.40gb" = "2g.10gb"
EOF
```

Quando i nodi dotati di queste impostazioni si uniscono al cluster, la modalità MIG viene abilitata sulle GPU, ogni GPU viene partizionata in `2g.10gb` istanze e il plug-in del dispositivo pubblicizza le istanze risultanti come risorsa. `nvidia.com/gpu` Poiché a `p4d.24xlarge` ha otto GPU A100 e ciascuna supporta tre istanze, il nodo fa pubblicità. `2g.10gb` `nvidia.com/gpu: 24`

> **Note**
>
> L'`mig.profile`impostazione è codificata in base al modello di GPU, ad esempio o. `a100.40gb` `h100.80gb` Senza un'`mig.profile`impostazione, la GPU attiva la modalità MIG e utilizza il suo profilo più grande. Poiché Bottlerocket utilizza la strategia singola, ogni

GPU sul nodo utilizza lo stesso profilo. Per utilizzare profili diversi sullo stesso nodo (strategia mista), utilizza il percorso AL2023 con NVIDIA GPU Operator.

### Usa MIG sui nodi AL2023 con il plug-in del dispositivo NVIDIA

In AL2023, i passaggi seguenti utilizzano NVIDIA GPU Operator per installare il plug-in del dispositivo NVIDIA e il MIG Manager. Il MIG Manager abilita la modalità MIG e partiziona le GPU in base alla configurazione fornita. Il plug-in del dispositivo NVIDIA quindi annuncia le istanze risultanti a Kubernetes. L'operatore GPU supporta sia la strategia singola che quella mista.

Poiché la NVIDIA AMI EKS-optimized AL2023 include già il driver e il toolkit NVIDIA, disabilita la gestione dei driver nel GPU Operator per evitare conflitti con il driver preinstallato. In alternativa, puoi installare e gestire tu stesso il plug-in del dispositivo NVIDIA e MIG Manager senza utilizzare l'operatore GPU.

#### Prerequisiti

- Un cluster Amazon EKS. La seguente procedura prevede il MIG-capable P-family provisioning di nodi (ad esempio`p4d.24xlarge`) con l'AMI NVIDIA EKS-optimized AL2023.
- Karpenter è installato e configurato nel cluster, perché la procedura crea un Karpenter `EC2NodeClass` e `NodePool` esegue il provisioning dei nodi GPU. Per ulteriori informazioni, consulta la Guida introduttiva a Karpenter sul sito Web di Karpenter.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurato per comunicare con il tuo cluster, vedi the section called "`Installa o aggiorna kubectl`" per ulteriori informazioni.

#### Procedura

1. Crea un `EC2NodeClass` and `NodePool` per i nodi P-family GPU AL2023. In AL2023, il partizionamento MIG viene applicato dall'operatore GPU nei passaggi successivi, quindi si tratta di una classe di nodi GPU AL2023 standard. L'esempio seguente esegue il provisioning dei nodi con l'AMI `p4d.24xlarge` NVIDIA AL2023. EKS-optimized

```
cat <<EOF | kubectl apply -f -
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
```

```
metadata:
  name: gpu-mig-al2023
spec:
  amiFamily: AL2023
  amiSelectorTerms:
    - alias: al2023@latest
  role: eksctl-KarpenterNodeRole-<cluster-name>
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: <cluster-name>
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: <cluster-name>
  tags:
    karpenter.sh/discovery: <cluster-name>
---
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-mig-al2023
spec:
  template:
    spec:
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: gpu-mig-al2023
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      requirements:
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["spot", "on-demand"]
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["p4d.24xlarge"]
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
  limits:
    cpu: 1000
    memory: 5000Gi
```

```
EOF
```

2. Aggiungi il repository NVIDIA Helm.

```
helm repo add nvidia https://nvidia.github.io/gpu-operator
helm repo update
```

3. Crea un `gpu-operator-values.yaml` file che disabiliti la gestione dei driver, selezioni la strategia mista e definisca i profili MIG da applicare. L'esempio seguente definisce una `p4d-half-balanced` configurazione che partiziona quattro delle otto GPU su un `p4d.24xlarge` nodo e lascia il resto intero.

```
cat <<EOF > gpu-operator-values.yaml
driver:
  enabled: false
toolkit:
  enabled: false
devicePlugin:
  enabled: true
nfd:
  enabled: true
gfd:
  enabled: true
mig:
  strategy: mixed
migManager:
  enabled: true
  env:
    - name: WITH_REBOOT
      value: "true"
  config:
    create: true
    name: custom-mig-parted-configs
    default: all-disabled
    data:
      config.yaml: |-
        version: v1
        mig-configs:
          all-disabled:
            - devices: all
              mig-enabled: false
          p4d-half-balanced:
            - devices: [0, 1, 2, 3]
```

```
              mig-enabled: true
              mig-devices:
                "1g.5gb": 2
                "2g.10gb": 1
                "3g.20gb": 1
            - devices: [4, 5, 6, 7]
              mig-enabled: false
EOF
```

4. Installa l'operatore GPU con il file dei valori.

```
helm install gpu-operator nvidia/gpu-operator \
    --namespace gpu-operator \
    --create-namespace \
    --values gpu-operator-values.yaml
```

5. Etichetta i tuoi MIG-capable nodi con la configurazione del profilo da applicare. Il componente MIG Manager controlla questa etichetta e partiziona le GPU di conseguenza, riavviando il nodo per applicare la modifica.

```
kubectl label nodes -l node.kubernetes.io/instance-type=p4d.24xlarge \
    nvidia.com/mig.config=p4d-half-balanced --overwrite
```

6. Dopo che l'operatore GPU ha partizionato le GPU, i Pods richiedono un profilo MIG specifico in base al nome della risorsa anziché. `nvidia.com/gpu` L'esempio seguente esegue un Pod che richiede un'istanza. `1g.5gb`

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: mig-inference
spec:
  tolerations:
    - key: nvidia.com/gpu
      operator: Exists
      effect: NoSchedule
  containers:
    - name: cuda
      image: nvidia/cuda:12.6.0-base-ubuntu22.04
      command: ["nvidia-smi", "-L"]
      resources:
```

```
        limits:
          nvidia.com/mig-1g.5gb: 1
EOF
```

Per esempi completi di configurazione a strategia mista per P-family le istanze, inclusi gruppi di nodi gestiti con prenotazioni di capacità e carichi di lavoro per profilo, consulta la sezione MIG della Amazon EKS Best Practices Guide. https://docs.aws.amazon.com/eks/latest/best-practices/aiml-compute.html

Verifica che MIG sia attivo

Dopo aver installato i MIG-enabled nodi`Ready`, verificate che la modalità MIG sia attiva sulle GPU e che il nodo pubblicizzi le risorse MIG previste.

1. Verifica che il nodo pubblicizzi le risorse MIG. Con la strategia unica, il nodo riporta le istanze come. `nvidia.com/gpu` Con la strategia mista, il nodo riporta le risorse specifiche del profilo, ad esempio. `nvidia.com/mig-1g.10gb`

   ```
   kubectl describe node <node-name> | grep nvidia.com
   ```

2. Esegui `nvidia-smi` da un Pod su un MIG-enabled nodo per confermare che la modalità MIG è attiva.

   `nvidia-smi`segnala `MIG M.: Enabled` le GPU con la modalità MIG attivata ed elenca le istanze MIG configurate su ciascuna. Di seguito è riportato un esempio di output da una GPU A100 da 40 GB con MIG abilitato e partizionata in, e istanze. `3g.20gb` `2g.10gb` `1g.5gb`

   ```
   +-----------------------------------------------------------------------------------------
   +
   | NVIDIA-SMI 580.159.03             Driver Version: 580.159.03     CUDA Version: 13.0
        |
   +-----------------------------------------+------------------------
   +----------------------+
   | GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
    ECC |
   | Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util
    Compute M. |
   |                                         |                        |
    MIG M. |
   |=========================================+========================
   +======================|
   ```

```
|   0  NVIDIA A100-SXM4-40GB          On  |   00000000:10:1C.0 Off |
  On |
| N/A   35C    P0             65W /  400W |     213MiB /  40960MiB |     N/A
 Default |
|                                         |                        |
 Enabled |
+-----------------------------------------+------------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| MIG devices:
     |
+------------------+----------------------------------+-----------
+-----------------------+
| GPU  GI  CI  MIG |              Shared Memory-Usage |        Vol|        Shared
     |
|      ID  ID  Dev |                Shared BAR1-Usage | SM     Unc| CE ENC  DEC  OFA
 JPG |
|                  |                                 |        ECC|
     |
|==================+==================================+===========
+=======================|
|  0    1   0   0  |            107MiB / 20096MiB    | 42      0 |  3   0    2    0
   0 |
|                  |              0MiB / 12211MiB    |           |
     |
+------------------+----------------------------------+-----------
+-----------------------+
|  0    5   0   1  |             71MiB /  9984MiB    | 28      0 |  2   0    1    0
   0 |
|                  |              0MiB /  6105MiB    |           |
     |
+------------------+----------------------------------+-----------
+-----------------------+
|  0   13   0   2  |             36MiB /  4864MiB    | 14      0 |  1   0    0    0
   0 |
|                  |              0MiB /  3052MiB    |           |
     |
+------------------+----------------------------------+-----------
+-----------------------+
```

Con la strategia mista, un Pod vede solo l'istanza MIG richiesta, non il layout completo della GPU del nodo. Il `mig-inference` Pod del passaggio precedente ha richiesto un'`nvidia.com/mig-1g.5gb`istanza, quindi `nvidia-smi -L` all'interno di quel Pod è elencato un singolo dispositivo MIG.

```
kubectl logs mig-inference
```

Di seguito viene riportato un output di esempio.

```
GPU 0: NVIDIA A100-SXM4-40GB (UUID: GPU-5b7ce860-1951-004c-4881-6ac6997df770)
  MIG 1g.5gb      Device  0: (UUID: MIG-9b065868-21b6-5b8d-8ab9-e99089ed472c)
```

Per vedere il layout completo delle partizioni di ogni GPU su un nodo, `nvidia-smi -L` esegui sull'host anziché all'interno di un Pod di carico di lavoro. Il comando seguente avvia un pod di debug privilegiato sul nodo ed esegue quello dell'host. `nvidia-smi` Sostituiscilo *`node-name`* con il nome del tuo MIG-enabled nodo.

```
kubectl debug node/<node-name> -it --profile=sysadmin --image=nvidia/cuda:12.6.0-
base-ubuntu22.04 -- chroot /host nvidia-smi -L
```

Di seguito è riportato un esempio di output per la `p4d-half-balanced` configurazione. Le prime quattro GPU sono partizionate in istanze MIG e le restanti quattro sono GPU intere. Ogni `MIG` riga è un'istanza isolata dall'hardware con UUID, memoria e slice di calcolo propri.

```
GPU 0: NVIDIA A100-SXM4-40GB (UUID: GPU-5b7ce860-1951-004c-4881-6ac6997df770)
  MIG 3g.20gb     Device  0: (UUID: MIG-7dc16162-7ba2-5894-abde-d753dc8ecf56)
  MIG 2g.10gb     Device  1: (UUID: MIG-56cfe1f0-0662-50e4-a5f7-111107e4d5e6)
  MIG 1g.5gb      Device  2: (UUID: MIG-1409727e-2ffa-5fd4-9586-204c9e2b36d5)
  MIG 1g.5gb      Device  3: (UUID: MIG-9b065868-21b6-5b8d-8ab9-e99089ed472c)
GPU 1: NVIDIA A100-SXM4-40GB (UUID: GPU-73692a43-dd2d-f1a1-b0df-2f4734e2a87d)
  MIG 3g.20gb     Device  0: (UUID: MIG-745fd93a-9582-57a6-8b3b-9782d289ca1b)
  MIG 2g.10gb     Device  1: (UUID: MIG-8440294e-c4a0-5687-add5-2cc506babb2f)
  MIG 1g.5gb      Device  2: (UUID: MIG-559810c0-f2bb-5ca2-b529-47228d99437f)
  MIG 1g.5gb      Device  3: (UUID: MIG-a91cf3f9-6459-59e9-97a8-dc69b9958eef)
GPU 2: NVIDIA A100-SXM4-40GB (UUID: GPU-4e56019e-84de-eef5-5ac3-85e468e93639)
  MIG 3g.20gb     Device  0: (UUID: MIG-c130392b-fd7c-59f8-9f4b-ecab27a2984b)
  MIG 2g.10gb     Device  1: (UUID: MIG-7d61cf16-ad08-57be-b2c2-ac6e515b28c3)
  MIG 1g.5gb      Device  2: (UUID: MIG-f5388dec-d841-506c-bb7a-6ac136ceee53)
  MIG 1g.5gb      Device  3: (UUID: MIG-02d54020-c6cb-5997-894b-e95af0f49388)
```

```
GPU 3: NVIDIA A100-SXM4-40GB (UUID: GPU-4fd894a0-b471-9e77-eb67-0ad15002ed5b)
  MIG 3g.20gb     Device  0: (UUID: MIG-10399b59-2625-5106-b3f8-76ae19da46e1)
  MIG 2g.10gb     Device  1: (UUID: MIG-ef81ee7d-fc48-56ed-8479-8ffa76eb4154)
  MIG 1g.5gb      Device  2: (UUID: MIG-ed9bf59d-6bf2-5e07-80fb-dd0d7ca41f9d)
  MIG 1g.5gb      Device  3: (UUID: MIG-a15d6f7d-661b-514e-8838-06fe4ffe7f75)
GPU 4: NVIDIA A100-SXM4-40GB (UUID: GPU-05b6b91b-da6e-3078-1f4f-a7bbf1ff7ed2)
GPU 5: NVIDIA A100-SXM4-40GB (UUID: GPU-078a8df1-f387-0315-6b0b-af12e082f6d5)
GPU 6: NVIDIA A100-SXM4-40GB (UUID: GPU-6cdeffe7-45f1-7e8e-bcc1-4634399ad877)
GPU 7: NVIDIA A100-SXM4-40GB (UUID: GPU-5f68814a-4e4a-5dec-79b4-8d70a61c7714)
```

## Usa il time-slicing con le GPU NVIDIA su Amazon EKS

Time-slicing consente a più Pod di condividere un'unica GPU fisica NVIDIA. Lo scheduler Kubernetes colloca più Pod sulla stessa GPU e lo scheduler CUDA della GPU multiplica il lavoro nel tempo. Time-slicing è la strategia più semplice. GPU-sharing Utilizza solo software, non richiede hardware speciale e funziona su tutti i tipi di istanze GPU NVIDIA. AWS Time-slicing non offre l'isolamento della memoria o del calcolo tra pod che condividono una GPU. Si adatta meglio ai carichi di lavoro con un basso utilizzo della GPU, come i servizi di inferenza che rimangono inattivi tra le richieste o gli ambienti di sviluppo in cui più utenti condividono una GPU. Per i carichi di lavoro che richiedono memoria a livello hardware e isolamento di elaborazione, utilizza invece la GPU (MIG). Multi-Instance

Su Amazon EKS, puoi gestire il time-slicing con il driver NVIDIA DRA o il plug-in del dispositivo NVIDIA. ???

Time-slicing può essere utilizzato con Karpenter solo se si utilizza il provisioning statico della capacità. https://karpenter.sh/docs/concepts/nodepools/#static-nodepool Time-slicing non è attualmente disponibile in modalità EKS Auto.

Time-slicing è adatto quando:

- L'utilizzo della GPU è costantemente basso, ad esempio i servizi di inferenza sensibili alla latenza che rimangono inattivi tra una richiesta e l'altra.
- Diversi sviluppatori condividono un singolo nodo GPU per il notebook o il lavoro di sviluppo.
- I nodi utilizzano un tipo di istanza GPU che non supporta MIG, ad esempio le famiglie `g5g6`, o. `g6e`
- Puoi accettare che un Pod sia occasionalmente più lento perché un altro Pod sulla stessa GPU è occupato.

Prendi in considerazione un approccio diverso quando:

- È necessario l'isolamento della memoria. I pod su una GPU frazionata condividono la stessa memoria della GPU e un Pod può esaurire la memoria su cui fanno affidamento gli altri Pod. Usa MIG per l'isolamento della memoria.
- È necessaria una latenza per POD o una qualità del servizio prevedibili. Lo scheduler GPU condivide l'elaborazione tra gli slot nel migliore dei modi e senza garanzie.
- Esegui carichi di lavoro di formazione. Time-slicing aggiunge il cambio di contesto che riduce l'efficienza della formazione. Un processo sottoposto a checkpoint rallentato da un altro Pod deve riprovare dall'ultimo checkpoint.

### Considerazioni

Esamina le seguenti considerazioni prima di utilizzare il time-slicing in produzione.

#### Considerazioni generali

- Nessun isolamento della memoria:  i pod che condividono una GPU suddivisa in base al tempo condividono la stessa memoria. Un Pod può allocare la memoria necessaria ad altri Pod, il che può causare errori di esaurimento della memoria. Abbina il numero di Pod che condividono ciascuna GPU all'ingombro di memoria dei tuoi carichi di lavoro. Se i tuoi Pod caricano regolarmente modelli di grandi dimensioni, condividi meno Pod per GPU o usa MIG.
- Best-effort condivisione dell'elaborazione: lo scheduler GPU condivide  l'elaborazione tra i Pod nel migliore dei modi e non garantisce un calcolo proporzionale per ciascun Pod.
- Time-slicing e MPS non può condividere la stessa GPU:  Time-slicing imposta la modalità di elaborazione della GPU su, mentre NVIDIA Service (MPS) lo richiede. `DEFAULT` Multi-Process `EXCLUSIVE_PROCESS` È possibile utilizzare entrambe le strategie nello stesso cluster, ma non sulla stessa GPU fisica contemporaneamente. Time-slicing inoltre non ha effetto su un'istanza MIG. Per condividere una singola istanza MIG tra contenitori, utilizzate invece MPS.
- Per-container metriche:  NVIDIA Data Center GPU Manager (DCGM) non può attribuire metriche ai singoli contenitori quando il time-slicing è attivo. GPU-level le metriche rimangono disponibili, ma non è possibile identificare quale Pod abbia consumato una determinata quantità di risorse GPU.

#### considerazioni sui driver NVIDIA DRA

- Funzione alfa:  Time-slicing tramite il driver DRA è necessaria la `TimeSlicingSettings` feature gate, che è una funzionalità alfa disattivata per impostazione predefinita. Per ulteriori informazioni, consulta the section called “Usa il time-slicing della GPU con il driver NVIDIA DRA”.

- User-mediated solo condivisione: i pod condividono una GPU solo facendo riferimento alla stessa `ResourceClaim` o`ResourceClaimTemplate`, con ambito di namespace, quindi la condivisione non può attraversare namespace. System-mediated la condivisione è una capacità futura proposta.
- Disabilita il plug-in del dispositivo integrato su Bottlerocket: il driver DRA non può essere eseguito insieme al plug-in del dispositivo NVIDIA sullo stesso nodo. Su Bottlerocket, disabilita il plug-in del dispositivo integrato, che richiede la versione Bottlerocket 1.63.0 o successiva. Per ulteriori informazioni, consulta the section called "Installa il driver NVIDIA DRA".
- Supporto di elaborazione: il driver NVIDIA DRA è supportato con il provisioning della capacità statica in Karpenter, gruppi di nodi gestiti da EKS o nodi autogestiti e non è supportato con EKS Auto Mode. Per ulteriori informazioni, consulta la documentazione statica di Karpenter sul sito Web di Karpenter. NodePool

Considerazioni sui plug-in dei dispositivi NVIDIA

- Best-effort condivisione del computer con slot: il plug-in del dispositivo pubblicizza un numero fisso di slot per GPU. Se un singolo Pod richiede più di uno slot, non riceve elaborazione aggiuntiva. Abilita l'`fail-requests-greater-than-one`opzione per rifiutare i Pod che richiedono più di uno slot.
- Provisioning con Karpenter: Karpenter conta ogni `nvidia.com/gpu` richiesta come GPU fisica anche quando il time-slicing è abilitato sulla GPU. Per ulteriori informazioni, consulta il numero #2140 di Karpenter su. https://github.com/kubernetes-sigs/karpenter/issues/2140 GitHub
- Nessun supporto per la modalità EKS Auto: EKS Auto Mode gestisce il plug-in del dispositivo NVIDIA e non ne espone la configurazione. Poiché il time-slicing richiede la configurazione del plug-in del dispositivo, non è possibile applicare una configurazione time-slicing sui nodi EKS Auto Mode.
- Time-slicing modifiche alla configurazione: il plug-in del dispositivo NVIDIA non monitora il time-slicing per eventuali modifiche. `ConfigMap` Riavvia il plug-in del dispositivo dopo aver aggiornato la configurazione.

Opzioni di configurazione

È possibile utilizzare il time-slicing per le GPU NVIDIA con le AMI NVIDIA EKS-optimized AL2023 e Bottlerocket. Le impostazioni del time-slicing variano a seconda che si stia utilizzando il driver NVIDIA DRA o il plug-in del dispositivo NVIDIA.

NVIDIA DRA driver

Le AMI EKS-optimized accelerate non includono il driver NVIDIA DRA. Se si utilizza Bottlerocket, è necessario disabilitare il plug-in del dispositivo NVIDIA integrato prima di utilizzare il driver NVIDIA DRA impostando i dati utente del nodo, che richiede Bottlerocket versione `settings.kubelet-device-plugins.nvidia.enabled = false` 1.63.0 o successiva. Per ulteriori informazioni, consulta the section called "Installa il driver NVIDIA DRA".

Con il driver NVIDIA DRA, il time-slicing è configurato tramite. ResourceClaimTemplates Il `interval` campo controlla la durata del time-slice CUDA.

| Interval | Description |
| --- | --- |
| Predefinita | Utilizza l'intervallo predefinito integrato nel driver GPU NVIDIA |
| Breve | Intervallo più breve; i contesti cambiano più frequentemente |
| Media | Intervallo intermedio |
| Long | ogni contesto dura più a lungo per turno prima di essere annullato |

NVIDIA device plugin

- Bottlerocket: l'AMI include un plug-in per dispositivi NVIDIA preinstallato. Puoi configurare il time-slicing tramite le impostazioni di Bottlerocket, senza l'installazione separata del plug-in del dispositivo, del diagramma Helm o. `ConfigMap`
- AL2023 — Si installa il plug-in del dispositivo NVIDIA come descritto in the section called "Installa il plug-in del dispositivo NVIDIA Kubernetes" e si fornisce la configurazione del time-slicing tramite un. `ConfigMap`

  Le seguenti opzioni controllano il modo in cui il plug-in del dispositivo NVIDIA pubblicizza e gestisce le GPU time-sliced. I nomi dei campi differiscono tra le impostazioni Bottlerocket e AL2023`ConfigMap`, come mostrato nelle procedure che seguono.

| Opzione | Valore consigliato | Description |
| --- | --- | --- |
| Repliche | 2—8 | Il numero di slot programmabili da pubblicizzare per GPU fisica. Valori più alti consentono una maggiore condivisione ma aumentano la contesa tra i Pod. |
| Rinomina per impostazione predefinita | `false` | Quando`false`, richiesto dai Pods`nvidia.com/gpu` , il che mantiene la compatibilità con i manifesti esistenti . Quando`true`, il plug-in del dispositivo pubblicizza la risorsa come `nvidia.com/gpu.shared` e Pods deve richiedere quel nome. Impostalo su `true` quando esegui pool di nodi GPU condivisi e dedicati e desideri che i carichi di lavoro selezionino esplicitamente la risorsa condivisa. |
| Richieste con esito negativo: più di una | `true` | Rifiuta i pod che richiedono più di uno slot suddiviso in base al tempo. Un Pod che richiede più di uno slot non riceve un calcolo proporzionale. Abilita questa impostazione per evitare un errore di configurazione comune. |

Per l'elenco completo delle opzioni di condivisione della GPU e dei relativi valori predefiniti, consulta la documentazione del plug-in per dispositivi NVIDIA Kubernetes su. GitHub

Usa il time-slicing della GPU con il driver NVIDIA DRA

Con il driver NVIDIA DRA, i Pod condividono una GPU facendo riferimento a una comune che la richiede con un time-slicing. `ResourceClaim` `gpu.nvidia.com` `DeviceClass` `GpuConfig`

Il driver NVIDIA DRA attualmente implementa il time-slicing mediato dall'utente: una GPU è condivisa solo tra i Pod e i contenitori a cui si fa esplicitamente riferimento alla stessa dichiarazione. Poiché a `ResourceClaim` e a hanno un ambito di namespace, `ResourceClaimTemplate` i Pod

che condividono una GPU in questo modo devono trovarsi nello stesso namespace. Per condividere una GPU tra contenitori in un singolo pod, fai in modo che ogni contenitore faccia riferimento allo stesso nome di richiesta nel claim. I contenitori che fanno riferimento a nomi di richiesta diversi ricevono ciascuno una GPU separata. Creane uno separato `ResourceClaimTemplate` per ogni intervallo di tempo necessario e uno in ogni namespace in cui desideri condividere le GPU.

System-mediatedil time-slicing si verifica quando il driver condivide una GPU tra claim indipendenti (inclusi namespace) in base a criteri definiti dal sistema anziché a un claim configurato dall'utente. Per ulteriori informazioni sul time-slicing mediato dal sistema, vedi System-mediated time-slicing of GPU (issue #659) e pull request #1257 on. https://github.com/kubernetes-sigs/dra-driver-nvidia-gpu/pull/1257 GitHub

> **Important**
>
> Time-slicing tramite il driver DRA richiede il `TimeSlicingSettings` feature gate, che è una funzionalità alfa disattivata per impostazione predefinita. Se richiedi la strategia di `TimeSlicing` condivisione senza abilitare questa funzionalità, il driver non riesce a preparare il dispositivo e il Pod rimane aggiornato `ContainerCreating` con un `FailedPrepareDynamicResources` evento che segnala`error validating GPU config: unknown GPU sharing strategy: TimeSlicing`. Attiva il feature gate solo se accetti i rischi derivanti dall'utilizzo di una funzionalità alfa.

Prerequisiti

- Un cluster Amazon EKS che esegue Kubernetes versione 1.34 o successiva. Il driver NVIDIA DRA è supportato con il provisioning statico della capacità in Karpenter, gruppi di nodi gestiti EKS o nodi autogestiti.
- Nodi con tipi di istanze GPU NVIDIA che utilizzano l'AMI NVIDIA AL2023. EKS-optimized
- Il driver NVIDIA DRA è stato installato come descritto inthe section called "Installa il driver NVIDIA DRA", con il feature gate abilitato. `TimeSlicingSettings`

Procedura

1. Crea un file `ResourceClaim` che richieda una GPU con la `TimeSlicing` strategia di condivisione. Più pod che fanno riferimento a questa affermazione condividono la stessa GPU fisica.

```
cat <<EOF | kubectl apply -f -
apiVersion: resource.k8s.io/v1
kind: ResourceClaim
metadata:
  name: shared-timeslice-gpu
spec:
  devices:
    requests:
    - name: gpu
      exactly:
        deviceClassName: gpu.nvidia.com
        count: 1
    config:
    - requests: ["gpu"]
      opaque:
        driver: gpu.nvidia.com
        parameters:
          apiVersion: resource.nvidia.com/v1beta1
          kind: GpuConfig
          sharing:
            strategy: TimeSlicing
            timeSlicingConfig:
              interval: Long
EOF
```

2. Distribuisci due o più Pod che fanno riferimento al nome condiviso. `ResourceClaim` Ogni Pod fa riferimento al reclamo tramite `resourceClaims` e. `resources.claims`

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: share-a
spec:
  tolerations:
    - key: nvidia.com/gpu
      operator: Exists
      effect: NoSchedule
  containers:
    - name: cuda
      image: nvidia/cuda:12.6.0-base-ubuntu22.04
      command: ["nvidia-smi", "-L"]
```

```
      resources:
        claims:
        - name: gpu
  resourceClaims:
    - name: gpu
      resourceClaimName: shared-timeslice-gpu
  restartPolicy: OnFailure
EOF
```

3. Verifica che i Pod condividano la stessa GPU fisica. Esegui `nvidia-smi -L` in ogni Pod e conferma che riportino lo stesso UUID della GPU.

```
kubectl logs share-a
```

Di seguito viene riportato un output di esempio. Un secondo Pod che fa riferimento alla stessa affermazione riporta un GPU UUID identico, il che conferma che entrambi i Pod condividono una GPU fisica.

```
GPU 0: NVIDIA L4 (UUID: GPU-b41973ee-5d0a-cde8-6287-12b53f861f02)
```

Usa il time-slicing della GPU sui nodi Bottlerocket con il plug-in per dispositivi NVIDIA

Su Bottlerocket, l'AMI accelerata include il plug-in del dispositivo NVIDIA. EKS-optimized Si abilita la suddivisione temporale `settings.kubelet-device-plugins.nvidia` delle impostazioni, che Bottlerocket trasforma nella configurazione del plug-in del dispositivo all'avvio del nodo.

Prerequisiti

- Un cluster Amazon EKS. La seguente procedura fornisce ai nodi GPU NVIDIA l'AMI NVIDIA EKS-optimized Bottlerocket.
- Karpenter è stato installato e configurato nel cluster, poiché la procedura seguente utilizza un Karpenter per fornire le impostazioni di time-slicing nei dati utente del nodo `EC2NodeClass` Bottlerocket. Per ulteriori informazioni, consulta la Guida introduttiva a Karpenter sul sito Web di Karpenter.
- `kubectl`configurato per comunicare con il tuo cluster, vedi the section called "`Installa o aggiorna kubectl`" per ulteriori informazioni.

Procedura

Aggiungi le impostazioni di time-slicing ai dati utente di Bottlerocket per i tuoi nodi GPU. L'esempio seguente configura quattro slot per GPU. Il modo in cui fornisci i dati utente dipende dal modo in cui esegui il provisioning dei nodi. L'esempio seguente mostra un Karpenter`EC2NodeClass`.

```
cat <<EOF | kubectl apply -f -
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-bottlerocket-timeslicing
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - alias: bottlerocket@latest
  role: eksctl-KarpenterNodeRole-<cluster-name>
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: <cluster-name>
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: <cluster-name>
  userData: |
    [settings.kubelet-device-plugins.nvidia]
    device-sharing-strategy = "time-slicing"

    [settings.kubelet-device-plugins.nvidia.time-slicing]
    replicas = 4
    rename-by-default = false
    fail-requests-greater-than-one = true
EOF
```

Quando i nodi dotati di queste impostazioni si uniscono al cluster, il plug-in del dispositivo annuncia quattro `nvidia.com/gpu` slot per ogni GPU fisica. Li richiedi nelle specifiche del carico di lavoro con le richieste di risorse del contenitore o i limiti per la risorsa estesa. `nvidia.com/gpu`

Usa il time-slicing della GPU sui nodi AL2023 con il plug-in per dispositivi NVIDIA

Su AL2023, installi il plug-in del dispositivo NVIDIA come descritto in e fornisci la configurazione del time-slicing in the section called "Installa il plug-in del dispositivo NVIDIA Kubernetes" un. `ConfigMap`

Prerequisiti

- Un cluster Amazon EKS con nodi che utilizzano i tipi di istanza GPU NVIDIA e l'AMI NVIDIA AL2023. EKS-optimized
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- `kubectl`configurato per comunicare con il tuo cluster, consulta the section called "`Installa o aggiorna kubectl`" per ulteriori informazioni.

Procedura

1. Crea il time-slicing `ConfigMap` nel `nvidia` namespace in cui hai installato il plug-in del dispositivo. the section called "Installa il plug-in del dispositivo NVIDIA Kubernetes" Questo esempio configura quattro slot per GPU.

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: nvidia-device-plugin-config
  namespace: nvidia
data:
  config.yaml: |
    version: v1
    sharing:
      timeSlicing:
        renameByDefault: false
        failRequestsGreaterThanOne: true
        resources:
          - name: nvidia.com/gpu
            replicas: 4
EOF
```

2. Aggiorna la versione esistente del plug-in del dispositivo NVIDIA per fare riferimento `ConfigMap` al valore. `config.name` Il `--reuse-values` flag conserva i valori impostati quando hai installato il plug-in del dispositivo in. the section called "Installa il plug-in del dispositivo NVIDIA Kubernetes"

```
helm upgrade nvdp nvdp/nvidia-device-plugin \
    --namespace nvidia \
    --reuse-values \
```

```
    --set config.name=nvidia-device-plugin-config
```

> **Note**
>
> Il plug-in del dispositivo non si ricarica automaticamente quando si modifica il. `ConfigMap` Dopo aver aggiornato la configurazione del time-slicing, riavvia il plug-in del dispositivo Pods per applicare la modifica.

Verifica che il time-slicing della GPU sia attivo

Una volta terminata la suddivisione temporale dei nodi`Ready`, verificate che il plug-in del dispositivo indichi il numero previsto di slot e che i Pod condividano una GPU fisica.

> **Note**
>
> Il controllo degli UUID condivisi in questa procedura dimostra più chiaramente il time-slicing quando i Pod atterrano sulla stessa GPU fisica. Su un nodo con una sola GPU, il plug-in del dispositivo pubblicizza quattro slot e tutti e quattro i Pod condividono la stessa GPU, quindi riportano lo stesso UUID. Su un nodo con più GPU fisiche, lo scheduler può posizionare i Pod su GPU diverse. Questi Pod riportano UUID diversi anche se il time-slicing è attivo. Per dimostrare la condivisione su una GPU, pianifica il carico di lavoro su un tipo di istanza a GPU singola. Ad esempio, aggiungi un selettore di nodi come `node.kubernetes.io/instance-type: g6.2xlarge` nella specifica Pod.

1. Verifica che il nodo indichi il numero configurato di slot GPU. Con quattro slot per GPU, viene segnalato un nodo con una GPU fisica. `4`

   ```
   kubectl get nodes "-o=custom-
   columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
   ```

   Di seguito viene riportato un output di esempio.

   ```
   NAME                                           GPU
   ip-192-168-11-225.us-west-2.compute.internal   4
   ```

2. Crea una distribuzione che esegua quattro repliche, ognuna delle quali richiede uno slot GPU.

```
cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: timeslicing-demo
spec:
  replicas: 4
  selector:
    matchLabels:
      app: timeslicing-demo
  template:
    metadata:
      labels:
        app: timeslicing-demo
    spec:
      tolerations:
        - key: nvidia.com/gpu
          operator: Exists
          effect: NoSchedule
      containers:
        - name: cuda
          image: nvidia/cuda:12.6.0-base-ubuntu22.04
          command: ["bash", "-c", "nvidia-smi --query-gpu=uuid --format=csv,noheader;
 sleep infinity"]
          resources:
            limits:
              nvidia.com/gpu: 1
EOF
```

3. Verifica che tutti e quattro i Pod siano pianificati sullo stesso nodo.

```
kubectl get pods -l app=timeslicing-demo -o wide
```

4. Conferma che tutti e quattro i Pod riportino lo stesso UUID della GPU. Un singolo UUID condiviso tra tutti e quattro i Pod conferma che una GPU fisica viene multiplexata nel tempo.

```
kubectl logs -l app=timeslicing-demo --prefix
```

   Di seguito viene riportato un output di esempio.

```
[pod/timeslicing-demo-xxxxxxxxxx-aaaaa/cuda] GPU-c0583cce-87c5-c736-db7f-6d3128c84d03
```

```
[pod/timeslicing-demo-xxxxxxxxxx-bbbbb/cuda] GPU-c0583cce-87c5-c736-db7f-6d3128c84d03
[pod/timeslicing-demo-xxxxxxxxxx-ccccc/cuda] GPU-c0583cce-87c5-c736-db7f-6d3128c84d03
[pod/timeslicing-demo-xxxxxxxxxx-ddddd/cuda] GPU-c0583cce-87c5-c736-db7f-6d3128c84d03
```

# Ricette per ottimizzare il tuo cluster Amazon EKS per i AI/ML carichi di lavoro

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Questa sezione è progettata per fornire semplici ricette per l'ottimizzazione del cluster Amazon EKS, in particolare per AI/ML carichi di lavoro che coinvolgono hardware specializzato. Troverai indicazioni su come impedire che i pod siano pianificati su nodi specifici aggiungendo taint ai gruppi di nodi gestiti, inclusi prerequisiti, procedure dettagliate e considerazioni sull'implementazione.

Argomenti


- Ricetta: impedire che i pod siano programmati su nodi specifici


## Ricetta: impedire che i pod siano programmati su nodi specifici

### Panoramica di

I nodi con processori specializzati, come le GPU, possono essere più costosi da eseguire rispetto ai nodi su macchine standard. Per proteggere questi nodi da carichi di lavoro che non richiedono hardware speciale, è possibile utilizzare taint Kubernetes. I taint contrassegnano i nodi per respingere i pod che non hanno tolleranze corrispondenti, assicurando che siano pianificati solo carichi di lavoro compatibili. Per ulteriori informazioni, consulta Taint e tolleranze nella documentazione Kubernetes.

I taint dei nodi Kubernetes possono essere applicati a gruppi di nodi gestiti nuovi ed esistenti utilizzando Console di gestione AWS o tramite l'API Amazon EKS. Questa ricetta mostra come applicare i taint ai gruppi di nodi gestiti di Amazon EKS utilizzando la CLI. AWS Per informazioni sulla creazione di un gruppo di nodi con un taint utilizzando il Console di gestione AWS, consulta. the section called "Crea"

## Prerequisiti

- Un cluster Amazon EKS esistente.
- AWS CLI installata e configurata con le autorizzazioni appropriate.

## Fasi

Passaggio 1: creazione di un gruppo di nodi con i taint

Utilizza il comando `aws eks create-nodegroup` per creare un nuovo gruppo di nodi gestito con taint. Questo esempio applica un taint con chiave `dedicated`, valore `gpuGroup` ed effetto `NO_SCHEDULE`.

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws: iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

Per ulteriori informazioni ed esempi, consulta taint nella documentazione di riferimento di Kubernetes.

Passaggio 2: aggiornamento dei taint su un gruppo di nodi esistente

Utilizzare il comando AWS CLI aws eks update-nodegroup-config per aggiungere, rimuovere o sostituire i taint per i gruppi di nodi gestiti.

```
aws eks update-nodegroup-config \
```

```
  --cluster-name my-cluster \
  --nodegroup-name node-taints-example \
  --taints 'removeTaints=[{key=dedicated,value=gpuGroup,effect=NO_SCHEDULE}]'
```

### Note

- I taint possono essere aggiornati dopo aver creato il gruppo di nodi utilizzando l'API `UpdateNodegroupConfig`.
- La chiave del taint deve iniziare con una lettera o un numero. Può includere lettere, numeri, trattini (`-`), punti (`.`) e trattini bassi (`_`). Può contenere fino a 63 caratteri.
- Facoltativamente, la chiave del taint può iniziare con un prefisso di sottodominio DNS e un singolo `/`. Se inizia con un prefisso di sottodominio DNS, può avere una lunghezza di 253 caratteri.
- Il valore è facoltativo e deve iniziare con una lettera o un numero. Può includere lettere, numeri, trattini (`-`), punti (`.`) e trattini bassi (`_`). Può contenere fino a 63 caratteri.
- Quando si utilizza Kubernetes direttamente o Console di gestione AWS, l'effetto taint deve essere `NoSchedule`, `PreferNoSchedule` o `NoExecute`. Tuttavia, quando si utilizza la AWS CLI o l'API, l'effetto contaminante deve essere`NO_SCHEDULE`,, o. `PREFER_NO_SCHEDULE` `NO_EXECUTE`
- Sono consentiti un massimo di 50 taint per un gruppo di nodi.
- Se i taint creati utilizzando un gruppo di nodi gestito vengono rimossi manualmente da un nodo, Amazon EKS non li aggiunge nuovamente al nodo. Questo è vero anche se i taint sono specificati nella configurazione del gruppo di nodi gestito.

# Risorse per iniziare a usare AI/ML Amazon EKS

Per passare al Machine Learning su EKS, iniziare scegliendo tra questi modelli prescrittivi per preparare rapidamente un cluster EKS e un software e un hardware ML per iniziare a eseguire carichi di lavoro ML.

## Workshop

### Workshop sull'IA generativa su Amazon EKS

Ulteriori informazioni su come iniziare a utilizzare le applicazioni e l'inferenza di un modello linguistico di grandi dimensioni (LLM) su Amazon EKS. Ulteriori informazioni su come implementare e gestire carichi di lavoro LLM di livello di produzione. Attraverso laboratori pratici, scoprirai come sfruttare Amazon EKS insieme a AWS servizi e strumenti open source per creare solide soluzioni LLM.

L'ambiente del workshop fornisce tutta l'infrastruttura e gli strumenti necessari, consentendo di rivolgere la propria attenzione sull'apprendimento e sull'implementazione.

### IA generativa su Amazon EKS con Neuron

Ulteriori informazioni su come iniziare a utilizzare le applicazioni e l'inferenza di un modello linguistico di grandi dimensioni (LLM) su Amazon EKS. Ulteriori informazioni su come implementare e gestire carichi di lavoro LLM di livello di produzione, implementare modelli RAG avanzati con database vettoriali e creare applicazioni LLM basate su dati utilizzando framework open source. Attraverso laboratori pratici, scoprirai come sfruttare Amazon EKS insieme a AWS servizi e strumenti open source per creare solide soluzioni LLM. L'ambiente del workshop fornisce tutta l'infrastruttura e gli strumenti necessari, consentendo di rivolgere la propria attenzione sull'apprendimento e sull'implementazione.

## Best practice

Gli AI/ML argomenti specifici della guida Amazon EKS Best Practices forniscono consigli dettagliati nelle seguenti aree per ottimizzare i AI/ML carichi di lavoro su Amazon EKS.

### AI/ML Calcolo e scalabilità automatica

Questa sezione descrive le best practice per l'ottimizzazione del AI/ML calcolo e della scalabilità automatica in Amazon EKS, concentrandosi sulla gestione delle risorse GPU, sulla resilienza dei nodi e sulla scalabilità delle applicazioni. Fornisce strategie come la pianificazione dei carichi di lavoro con etichette e affinità di nodi note, l'utilizzo di ML Capacity Blocks o On-Demand Capacity Reservations e l'implementazione di controlli dello stato dei nodi con strumenti come EKS Node Monitoring Agent.

### AI/ML Reti

Questa sezione descrive le migliori pratiche per ottimizzare la AI/ML rete in Amazon EKS per migliorare prestazioni e scalabilità, incluse strategie come la selezione di istanze con una larghezza di banda di rete maggiore o Elastic Fabric Adapter (EFA) per la formazione distribuita, l'installazione di strumenti come MPI e NCCL e l'abilitazione della delega dei prefissi per aumentare gli indirizzi IP e migliorare i tempi di avvio dei pod.

### AI/ML Sicurezza

Questa sezione si concentra sulla protezione dell'archiviazione dei dati e sulla garanzia della conformità per i AI/ML carichi di lavoro su Amazon EKS, incluse pratiche come l'utilizzo di

Amazon S3 AWS con Key Management Service (KMS) per la crittografia SSE-KMS lato server (), la configurazione di bucket con chiavi KMS regionali e chiavi S3 Bucket per ridurre i costi, la concessione di autorizzazioni IAM per azioni KMS come la decrittografia dei pod EKS e il controllo con i log. AWS CloudTrail

### AI/ML Archiviazione

Questa sezione fornisce le migliori pratiche per ottimizzare lo storage nei AI/ML carichi di lavoro su Amazon EKS, incluse pratiche come la distribuzione di modelli che utilizzano driver CSI per montare servizi come S3, FSx for Lustre o EFS come volumi persistenti, la selezione dello storage in base alle esigenze del carico di lavoro (ad esempio, FSx for Lustre per la formazione distribuita con Scratch-SSD opzioni come or) e l'abilitazione di funzionalità come la compressione e lo striping dei dati. Persistent-SSD

### AI/ML Osservabilità

Questa sezione si concentra sul monitoraggio e l'ottimizzazione dell'utilizzo della GPU per i AI/ML carichi di lavoro su Amazon EKS per migliorare l'efficienza e ridurre i costi, incluse strategie come mirare all'utilizzo elevato della GPU con strumenti CloudWatch come Container Insights e NVIDIA integrati DCGM-Exporter con Prometheus e Grafana, e metriche che consigliamo di analizzare per i carichi di lavoro. AI/ML

### AI/ML Prestazioni

Questa sezione si concentra sul miglioramento della scalabilità e delle prestazioni delle applicazioni per i AI/ML carichi di lavoro su Amazon EKS attraverso la gestione delle immagini dei container e l'ottimizzazione dell'avvio, incluse pratiche come l'utilizzo di immagini di base piccole e leggere o AWS Deep Learning Containers con build in più fasi, il precaricamento delle immagini tramite snapshot EBS o il preinserimento nella cache di runtime utilizzando o Deployments. DaemonSets

## Architetture di riferimento

Esplora questi GitHub repository per architetture di riferimento, codice di esempio e utilità per implementare formazione e inferenza distribuite per carichi di lavoro AI/ML su Amazon EKS e altri servizi. AWS

### Addestramento distribuito AWSome

Questo repository offre una raccolta di best practice, architetture di riferimento, esempi di addestramento dei modelli e utilità per addestrare modelli di grandi dimensioni su AWS. Supporta

la formazione distribuita con Amazon EKS, inclusi CloudFormation modelli per cluster EKS, build AMI e container personalizzate, casi di test per framework come PyTorch (DDP/FSDPMegatronLM NeMo) e JAX e strumenti per la convalida, l'osservabilità e il monitoraggio delle prestazioni come EFA Prometheus exporter e Nvidia Nsight Systems.

### Inferenza AWSome

Questo repository fornisce architetture di riferimento e casi di test per ottimizzare le soluzioni di inferenza AWS, con particolare attenzione ad Amazon EKS e alle istanze EC2 accelerate. Include configurazioni di infrastruttura per cluster VPC ed EKS, progetti per framework come NVIDIA NiMS, Triton Inference Server RayService e TensorRT-LLM, con esempi per modelli come Llama 3.1 405B. Llama3-8B Presenta implementazioni multinodo con K8s LeaderWorkerSet, scalabilità automatica EKS, GPU (MIG) e casi d'uso reali come un bot audio per ASR, inferenza e Multi-Instance TTS.

## Esercitazioni

Se si desidera configurare piattaforme e framework di Machine Learning in EKS, è necessario esplorare i tutorial descritti in questa sezione. Questi tutorial coprono tutto, dai modelli per sfruttare al meglio i processori delle GPU alla scelta degli strumenti di modellazione alla creazione di framework per settori specializzati.

### Creare piattaforme di IA generativa su EKS

- Implementazione dei modelli di IA generativa su Amazon EKS
- Creazione di JupyterHub piattaforme multi-tenant su Amazon EKS

### Eseguire framework di IA generativa specializzati su EKS

- Accelera i tuoi carichi di lavoro di formazione generativa distribuita con l'intelligenza artificiale con NVIDIA Framework NeMo su Amazon EKS
- TorchServe In esecuzione su Amazon Elastic Kubernetes Service

### Massimizzare le prestazioni della GPU NVIDIA per ML su EKS

- Implementare la condivisione delle GPU per utilizzare in modo efficiente le GPU NVIDIA per i propri cluster EKS:

  Condivisione della GPU su Amazon EKS con time-slicing NVIDIA e istanze EC2 accelerate

- Usa Multi-Instance GPU (MiGS) e microservizi NIM per eseguire più pod per GPU sui tuoi cluster EKS:

  Massimizzazione dell'utilizzo della GPU con la Multi-Instance GPU NVIDIA (MIG) su Amazon EKS: utilizzo di più pod per GPU per prestazioni migliorate
- Crea e distribuisci un sistema di machine learning scalabile su Kubernetes con Kubeflow on AWS

### Eseguire carichi di lavoro di codifica video su EKS

- Distribuzione di contenuti video con GPU frazionarie in container su Amazon EKS

### Accelerare il caricamento delle immagini per i carichi di lavoro di inferenza

- In che modo è stata H2O.ai ottimizzata e protetta la loro infrastruttura con Karpenter e Bottlerocket AI/ML

### Monitoraggio dei carichi di lavoro ML

- Monitoraggio dei carichi di lavoro delle GPU su Amazon EKS utilizzando servizi open source AWS gestiti
- Abilita le metriche GPU basate su pod in Amazon CloudWatch

# Strumenti Amazon EKS

Amazon EKS offre una suite di AWS strumenti che ti aiutano a gestire, risolvere i problemi e interagire con i tuoi cluster EKS in modo più efficiente. Questi strumenti sfruttano funzionalità basate sull'intelligenza artificiale e interfacce standardizzate per semplificare le operazioni dei cluster, accelerare la risoluzione dei problemi e consentire interazioni in linguaggio naturale con le risorse Kubernetes. Gli strumenti disponibili includono:

- Amazon EKS Model Context Protocol (MCP) Server: un servizio completamente gestito che consente esperienze basate sull'intelligenza artificiale per la gestione dei cluster EKS. Il server MCP EKS si integra con assistenti di codifica AI compatibili con MCP per fornire una conoscenza contestuale e in tempo reale dei cluster e delle risorse Kubernetes. Usalo per gestire i cluster, distribuire applicazioni, risolvere problemi e interrogare la documentazione EKS tramite interazioni in linguaggio naturale.
- Amazon Q sulla console Amazon EKS: un'integrazione per la risoluzione dei problemi basata sull'intelligenza artificiale disponibile direttamente nella console di AWS gestione. Amazon Q analizza automaticamente i problemi dei cluster e fornisce i risultati delle indagini, l'analisi della causa principale e le misure di mitigazione suggerite in caso di errori o avvisi nella console EKS.

Questi strumenti collaborano per fornire un supporto completo per tutto il ciclo di vita del cluster EKS, dalla configurazione e configurazione iniziali alle operazioni continue e alla risoluzione dei problemi. Che stiate sviluppando applicazioni, gestendo l'infrastruttura o risolvendo problemi di produzione, questi AWS strumenti vi aiutano a lavorare in modo più efficiente con i vostri cluster EKS.

Argomenti


- Server Amazon EKS Model Context Protocol (MCP)
- Usa Amazon Q Developer sulla console Amazon EKS


# Server Amazon EKS Model Context Protocol (MCP)

Il server MCP Amazon EKS è un servizio completamente gestito che consente AI-powered esperienze di sviluppo e operazioni. Il Model Context Protocol (MCP) fornisce un'interfaccia standardizzata che arricchisce gli agenti e le applicazioni AI con una conoscenza contestuale e in tempo reale dei cluster EKS e delle risorse Kubernetes, consentendo risposte e AI-powered

flussi di lavoro più accurati e sensibili al contesto durante tutto il ciclo di vita dell'applicazione, dalla configurazione iniziale all'ottimizzazione della produzione e alla risoluzione dei problemi.

> **Note**
>
> Il server MCP Amazon EKS è disponibile in anteprima per Amazon EKS ed è soggetto a modifiche.

## Panoramica di

Il server MCP EKS può essere facilmente integrato con qualsiasi assistente di codifica MCP-compatible AI per migliorare il flusso di lavoro di sviluppo come Kiro, Amazon Q Developer CLI o strumenti di terze parti come Cursor o Cline. https://cursor.com/ https://cline.bot/ All'inizio, il server EKS MCP ti guida nella creazione del cluster, assegnando automaticamente i prerequisiti e applicando le migliori pratiche. AWS Durante lo sviluppo, semplifica le operazioni di EKS e Kubernetes fornendo flussi di lavoro di alto livello per la distribuzione delle applicazioni e la gestione dei cluster. Per il debug e la risoluzione dei problemi, il server accelera la risoluzione dei problemi attraverso ausili integrati per la risoluzione dei problemi e l'accesso alla knowledge base disponibili sulla console EKS e sui tuoi assistenti AI preferiti. Queste funzionalità sono accessibili tramite interazioni in linguaggio naturale, consentendoti di eseguire operazioni Kubernetes complesse in modo più intuitivo ed efficiente.

Il server EKS MCP completamente gestito è ospitato nel AWS cloud, eliminando la necessità di installazione e manutenzione locali. Fornisce funzionalità di livello aziendale come aggiornamenti e patch automatici, sicurezza centralizzata tramite l'integrazione AWS IAM, registrazione completa degli audit via e la comprovata scalabilità AWS CloudTrail, affidabilità e supporto di. AWS Il server EKS MCP completamente gestito ospitato nel cloud offre i seguenti vantaggi chiave: AWS

- Elimina l'installazione e la manutenzione.  Con il server EKS MCP ospitato nel AWS cloud, non è più necessario gestire gli aggiornamenti delle versioni o risolvere i problemi del server locale. Basta configurare il tuo assistente AI per connettersi all'endpoint del server EKS MCP ospitato e sei pronto per iniziare a lavorare con i tuoi cluster EKS.
- Gestione centralizzata degli accessi.  Il server EKS MCP si integra con IAM, fornendo un modo centralizzato e sicuro per controllare l'accesso al server. Tutte le richieste vengono firmate utilizzando AWS SIGv4 tramite un proxy leggero, che consente una perfetta integrazione con le credenziali e le politiche IAM esistenti. AWS

- Monitoraggio e visibilità migliorati.  AWS CloudTrail l'integrazione acquisisce l'inizializzazione e l'accesso completo alle chiamate agli strumenti effettuate tramite il servizio in hosting, consentendo audit trail dettagliati e report di conformità.
- Sempre aggiornato.  Ricevi automaticamente nuovi strumenti, funzionalità e correzioni di bug senza dover aggiornare le installazioni locali. Il servizio in hosting viene continuamente migliorato in base al tuo feedback e alle AWS best practice.

## Esempi di integrazione

Il server EKS MCP fornisce diversi strumenti che è possibile utilizzare per:

- Gestisci il tuo cluster  Crea, configura e gestisci i cluster EKS con best practice automatizzate.
- Gestisci le risorse Kubernetes  Distribuisci applicazioni, gestisci le risorse Kubernetes e controlla lo stato del cluster.
- Risolvi i problemi del cluster  Diagnostica i problemi utilizzando strumenti integrati per la risoluzione dei problemi e la knowledge base dei runbook
- Documentazione delle query  Cerca e recupera la documentazione EKS pertinente contestualmente.

### Esplora i tuoi cluster

```
Show me all EKS clusters and their status
What insights does EKS have about my production-cluster?
Show me the VPC configuration for my staging cluster
```

### Consulta le risorse Kubernetes

```
List all deployments in the production namespace
Show me pods that are not in Running state
Get the logs from the api-server pod in the last 30 minutes
```

### Risoluzione dei problemi di

```
Why is my nginx-ingress-controller pod failing to start?
Search the EKS troubleshooting guide for pod networking issues
Show me events related to the failed deployment in the staging namespace
```

### Crea risorse (se la modalità «scrittura» è abilitata)

```
Create a new EKS cluster named demo-cluster with VPC and Auto Mode
Deploy my containerized app from ECR to the production namespace with 3 replicas
Generate a Kubernetes deployment YAML for my Node.js app running on port 3000
```

## Nozioni di base

Per iniziare, consulta the section called "Nozioni di base".

## Guida introduttiva al server MCP Amazon EKS

Questa guida illustra i passaggi per configurare e utilizzare il server MCP EKS con i tuoi assistenti di codice AI. Imparerai come configurare il tuo ambiente, connetterti al server e iniziare a gestire i tuoi cluster EKS tramite interazioni in linguaggio naturale.

> **Note**
>
> Il server MCP Amazon EKS è in versione di anteprima per Amazon EKS ed è soggetto a modifiche.

### Prerequisiti

Prima di iniziare, esegui le seguenti attività:

- Crea un AWS account con accesso ad Amazon EKS
- Installa e configura la AWS CLI con le credenziali
- Installa Python 3.10+
- Installa uv

### Configurazione

1. Verifica dei prerequisiti di

```
# Check that your Python version is 3.10 or higher
python3 --version
```

```
# Check uv installation
uv --version

# Verify CLI configuration
aws configure list
```

2. Configurare le autorizzazioni IAM

Per connettersi al server EKS MCP, il ruolo IAM deve avere le seguenti politiche allegate: **`eks-mcp:InvokeMcp`** (autorizzazioni richieste per l'inizializzazione e il recupero delle informazioni sugli strumenti disponibili), **`eks-mcp:CallReadOnlyTool`** (autorizzazioni richieste per l'utilizzo di strumenti di sola lettura) e (autorizzazioni richieste per l'utilizzo degli strumenti di accesso **`eks-mcp:CallPrivilegedTool`** completo (scrittura)). Queste `eks-mcp` autorizzazioni sono incluse nelle politiche gestite di sola lettura e accesso completo fornite di seguito. AWS

- Apri la console IAM.
- Nel riquadro di navigazione a sinistra, scegli Utenti , Gruppi di utenti o Ruoli a seconda dell'identità a cui desideri allegare la policy, quindi il nome dell'utente, del gruppo o del ruolo specifico.
- Scegli la scheda Autorizzazioni.
- Scegli Allega criteri (o Aggiungi autorizzazioni se è la prima volta).
- Nell'elenco delle policy, cerca e seleziona la policy gestita che desideri allegare:
- Read-only operazioni: AmazonEKSMCPReadOnlyAccess
- Scegli Allega criteri (o Avanti, quindi Aggiungi autorizzazioni per confermare).

Questo allega la policy e le autorizzazioni hanno effetto immediato. È possibile allegare più policy alla stessa identità e ognuna può contenere autorizzazioni diverse. Per ulteriori informazioni su queste politiche, consulta le politiche AWS gestite per Amazon Elastic Kubernetes Service.

3. Scegli un assistente AI

Scegli uno dei seguenti assistenti MCP-compatible AI o qualsiasi MCP-compatible strumento:

- Installa l'interfaccia a riga di comando per sviluppatori di Amazon Q
- Installa Kiro
- Installa Cursor
- Installa l'estensione del codice Cline VS

## Passaggio 1: configura il tuo assistente AI

Scegli tra una delle seguenti opzioni per configurare il tuo assistente al codice AI. Completando questo passaggio, configura il tuo assistente al codice AI per utilizzare il proxy MCP AWS, necessario per un accesso sicuro e autenticato al server MCP Amazon EKS. Ciò comporta l'aggiunta o la modifica del file di configurazione MCP (ad esempio, `~/.aws/amazonq/mcp.json` per Amazon Q Developer CLI). Il proxy funge da bridge sul lato client, gestendo l'autenticazione AWS SIGv4 utilizzando AWS le credenziali locali e consentendo il rilevamento dinamico degli strumenti per l'interazione con i server MCP di backend AWS come il server MCP EKS. Per saperne di più, consulta il proxy MCP per il repository. AWS

Opzione A: Amazon Q Developer CLI

L'interfaccia a riga di comando Q Developer offre l'esperienza più integrata con il server EKS MCP.

1. Individua il file di configurazione MCP

- macOS/Linux: `~/.aws/q/mcp.json`
- Windows: `%USERPROFILE%\.aws\q\mcp.json`

2. Aggiungi la configurazione del server MCP

Crea il file di configurazione se non esiste. Assicuratevi di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per: Mac/Linux

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
```

```
        "{region}"
      ]
    }
  }
}
```

Per Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Nota di sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni sugli strumenti di sola lettura.

3. Verifica la configurazione

Riavvia Q Developer CLI, quindi controlla gli strumenti disponibili:

```
q /tools
```

Opzione B: Kiro IDE

Kiro è uno spazio di lavoro di AI-first codifica con supporto MCP integrato. https://kiro.dev/docs/mcp/

1. Apri le impostazioni di Kiro

- Apri Kiro
- Vai su Kiro → Impostazioni e cerca «MCP Config»
- Oppure premi `Cmd+Shift+P` (Mac) o `Ctrl+Shift+P` (Windows/Linux) e cerca «MCP Config»

2. Aggiungi la configurazione del server MCP

- Fai clic su «Open Workspace MCP Config» o «Open User MCP Config» per modificare direttamente il file di configurazione MCP.

Assicurati di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per: Mac/Linux

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Per Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
```

```
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Nota di sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni sugli strumenti di sola lettura.

Opzione C: Cursor IDE

Cursor fornisce il supporto MCP integrato con un'interfaccia di configurazione grafica.

1. Apri le impostazioni del cursore

- Apri il cursore
- Vai su Impostazioni → Impostazioni del cursore → Strumenti e MCP
- Oppure premi `Cmd+Shift+P` (Mac)/`Ctrl+Shift+P`(Windows) e cerca «MCP»

2. Aggiungi la configurazione del server MCP

- Fai clic su «Nuovo server MCP»

Crea il file di configurazione se non esiste. Assicuratevi di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per: Mac/Linux

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Per Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
```

```
}
```

Nota di sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni sugli strumenti di sola lettura.

3. Cursore di riavvio

Chiudi e riapri Cursor per rendere effettive le modifiche.

4. Verifica nella chat di Cursor

Apri il pannello della chat e prova:

```
What EKS MCP tools are available?
```

Dovresti vedere un elenco degli strumenti di gestione EKS disponibili.

Opzione D: Cline (VS Code Extension)

Cline è una popolare estensione VS Code che porta l'assistenza AI direttamente nel tuo editor.

1. Apri le impostazioni di Cline

- Apri Cline
- Premi `Cmd+Shift+P` (Mac)/`Ctrl+Shift+P`(Windows) e cerca «MCP»

2. Aggiungi la configurazione del server MCP

- Fai clic su «Aggiungi server»
- Fai clic su «Apri configurazione utente»

Crea il file di configurazione se non esiste. Assicuratevi di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per: Mac/Linux

```
{
  "mcpServers": {
    "eks-mcp": {
```

```
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Per Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Nota di sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni sugli strumenti di sola lettura.

3. Ricarica VS Code

PremiCmd+Shift+P/Ctrl+Shift+Pe seleziona «Sviluppatore: Reload Window»

4. Verifica la configurazione

Apri Cline e chiedi:

```
List the available MCP tools for EKS
```

## Passaggio 2: (Facoltativo) Crea una politica di «scrittura»

Facoltativamente, puoi creare una policy IAM gestita dal cliente che fornisca l'accesso completo al server MCP Amazon EKS. Questa policy concede le autorizzazioni per utilizzare tutti gli strumenti del server EKS MCP, inclusi gli strumenti privilegiati che possono comportare operazioni di scrittura e gli strumenti di sola lettura. Tieni presente che le autorizzazioni ad alto rischio (qualsiasi cosa con Delete* o risorsa IAM senza restrizioni) sono incluse in questa policy, poiché sono necessarie per le risorse del cluster nello strumento manage_eks_stacks. setup/teardown

```
aws iam create-policy \
 --policy-name EKSMcpWriteManagementPolicy \
 --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect
\": \"Allow\", \"Action\": [\"eks:DescribeCluster\", \"eks:ListClusters\",
 \"eks:DescribeNodegroup\", \"eks:ListNodegroups\", \"eks:DescribeAddon\",
 \"eks:ListAddons\", \"eks:DescribeAccessEntry\", \"eks:ListAccessEntries\",
 \"eks:DescribeInsight\", \"eks:ListInsights\", \"eks:AccessKubernetesApi\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"eks:CreateCluster
\", \"eks:DeleteCluster\", \"eks:CreateAccessEntry\", \"eks:TagResource\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"iam:GetRole\",
 \"iam:ListRolePolicies\", \"iam:ListAttachedRolePolicies\", \"iam:GetRolePolicy\",
 \"iam:GetPolicy\", \"iam:GetPolicyVersion\"], \"Resource\": \"*\"}, {\"Effect\":
 \"Allow\", \"Action\": [\"iam:TagRole\", \"iam:CreateRole\", \"iam:AttachRolePolicy
\", \"iam:PutRolePolicy\", \"iam:DetachRolePolicy\", \"iam:DeleteRole\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"iam:PassRole\"],
 \"Resource\": \"*\", \"Condition\": {\"StringEquals\": {\"iam:PassedToService
\": [\"eks.amazonaws.com\", \"ec2.amazonaws.com\"]}}}, {\"Effect\": \"Allow\",
 \"Action\": [\"ec2:CreateVpc\", \"ec2:CreateSubnet\", \"ec2:CreateRouteTable\",
 \"ec2:CreateRoute\", \"ec2:CreateInternetGateway\", \"ec2:CreateNatGateway\",
 \"ec2:CreateSecurityGroup\", \"ec2:AttachInternetGateway\", \"ec2:AssociateRouteTable
\", \"ec2:ModifyVpcAttribute\", \"ec2:ModifySubnetAttribute\", \"ec2:AllocateAddress
\", \"ec2:CreateTags\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"ec2:DeleteVpc\", \"ec2:DeleteSubnet\", \"ec2:DisassociateRouteTable\",
```

```
 \"ec2:DeleteRouteTable\", \"ec2:DeleteRoute\", \"ec2:DetachInternetGateway\",
 \"ec2:DeleteInternetGateway\", \"ec2:DeleteNatGateway\", \"ec2:ReleaseAddress\",
 \"ec2:DeleteSecurityGroup\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"ec2:DescribeVpcs\", \"ec2:DescribeSubnets\", \"ec2:DescribeRouteTables\",
 \"ec2:DescribeInternetGateways\", \"ec2:DescribeNatGateways\", \"ec2:DescribeAddresses
\", \"ec2:DescribeSecurityGroups\", \"ec2:DescribeAvailabilityZones\"], \"Resource
\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"cloudformation:CreateStack
\", \"cloudformation:UpdateStack\", \"cloudformation:DeleteStack\",
 \"cloudformation:DescribeStacks\", \"cloudformation:TagResource\"], \"Resource
\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"sts:GetCallerIdentity\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"logs:StartQuery\",
 \"logs:GetQueryResults\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\":
 [\"cloudwatch:GetMetricData\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"eks-mcp:*\"], \"Resource\": \"*\"}]}"
```

## Passaggio 3: verifica la configurazione

### Connessione di prova

Fai una semplice domanda al tuo assistente AI per verificare la connessione:

```
List all EKS clusters in my {aws} account
```

Dovresti vedere un elenco dei tuoi cluster EKS.

## Fase 4: Esegui le tue prime attività

### Esempio 1: esplora i tuoi cluster

```
Show me all EKS clusters and their status
What insights does EKS have about my production-cluster?
Show me the VPC configuration for my staging cluster
```

### Esempio 2: controlla le risorse Kubernetes

```
Get the details of all the kubernetes resources deployed in my EKS cluster
Show me pods that are not in Running state or pods with any restarts
Get the logs from the aws-node daemonset in the last 30 minutes
```

### Esempio 3: risoluzione dei problemi

```
Why is my nginx-ingress-controller pod failing to start?
```

```
Search the EKS troubleshooting guide for pod networking issues
Show me events related to the failed deployment in the staging namespace
```

Esempio 4: creazione di risorse (se la modalità «scrittura» è abilitata)

```
Create a new EKS cluster named demo-cluster with VPC and Auto Mode
Deploy my containerized app from ECR to the production namespace with 3 replicas
Generate a Kubernetes deployment YAML for my Node.js app running on port 3000
```

## Configurazioni comuni

Scenario 1: multiplo AWS profili

Se lavorate con più AWS account, create configurazioni separate del server MCP.

Per: Mac/Linux

```
{
  "mcpServers": {
    "eks-mcp-prod": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.us-west-2.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2"
      ]
    },
    "eks-mcp-dev": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.us-east-1.api.aws/mcp",
        "--service",
        "eks-mcp",
```

```
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Per Windows:

```
{
  "mcpServers": {
    "eks-mcp-prod": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.us-west-2.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2"
      ]
    },
    "eks-mcp-dev": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.us-east-1.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
```

```
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Scenario 2: Read-only per la produzione

Crea una configurazione di sola lettura per gli ambienti di produzione.

Per: Mac/Linux

```
{
  "mcpServers": {
    "eks-mcp-prod-readonly": {
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.us-west-2.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2",
        "--read-only"
      ],
      "autoApprove": [
        "list_k8s_resources",
        "get_pod_logs",
        "get_k8s_events"
      ]
    }
  }
}
```

Per Windows:

```
{
  "mcpServers": {
    "eks-mcp-prod-readonly": {
      "command": "uvx",
```

```
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.us-west-2.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2",
        "--read-only"
      ],
      "autoApprove": [
        "list_k8s_resources",
        "get_pod_logs",
        "get_k8s_events"
      ]
    }
  }
}
```

Scenario 3: sviluppo con accesso completo

Per ambienti di sviluppo con accesso completo in scrittura.

Per Mac/Linux:

```
{
  "mcpServers": {
    "eks-mcp-dev-full": {
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.us-east-1.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
```

```
}
```

Per Windows:

```
{
  "mcpServers": {
    "eks-mcp-dev-full": {
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.us-east-1.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

## Considerazioni

### Sicurezza

Non trasmettere informazioni segrete o sensibili tramite meccanismi di input consentiti:

- Non includere segreti o credenziali nei file YAML applicati con apply_yaml.
- Non passate informazioni sensibili direttamente nel prompt al modello.
- Non includete segreti nei CloudFormation modelli o nei manifesti delle applicazioni.
- Evita di utilizzare strumenti MCP per creare Kubernetes Secrets, poiché ciò richiederebbe la fornitura dei dati segreti al modello.
- Evita di registrare informazioni sensibili nei log delle applicazioni all'interno dei pod Kubernetes.

Sicurezza dei contenuti YAML:

- Usa solo file YAML provenienti da fonti affidabili.

- Il server si basa sulla convalida dell'API Kubernetes per i contenuti YAML e non esegue la propria convalida.
- Controlla i file YAML prima di applicarli al tuo cluster.

Invece di trasmettere segreti tramite MCP:

- Usa AWS Secrets Manager o Parameter Store per archiviare informazioni sensibili.
- Configura il RBAC Kubernetes corretto per gli account di servizio.
- Usa i ruoli IAM per gli account di servizio (IRSA) per l'accesso ai servizi dai pod. AWS

Oscurazione dei dati sensibili:

- Il server EKS MCP oscura automaticamente i modelli più comuni relativi a token di sicurezza, certificati e altre informazioni sensibili nelle risposte degli strumenti.
- I valori oscurati vengono sostituiti con `HIDDEN_FOR_SECURITY_REASONS` per evitare l'esposizione accidentale dei dati al modello.
- Questa redazione si applica a tutte le risposte degli strumenti, inclusi i registri, le descrizioni delle risorse e i dati di configurazione.

### Prossimo

Per le opzioni di configurazione, vederethe section called "Configurazione". Per un elenco completo degli strumenti, vederethe section called "Tools (Strumenti)".

## Riferimento per la configurazione del server MCP Amazon EKS

Questa guida mostra tutte le configurazioni disponibili per lo strumento lato client mcp-proxy-for-aws che consente di connettersi al server MCP Amazon EKS completamente gestito dal tuo IDE.

> **Note**
>
> Il server MCP Amazon EKS è in versione di anteprima per Amazon EKS ed è soggetto a modifiche.

## Esempio

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.us-west-2.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "us-east-1",
        "--read-only"
      ]
    }
  }
}
```

## autorizzazioni IAM

Il ruolo utilizzato per la connessione al server MCP richiede **`eks-mcp:InvokeMcp`** autorizzazioni per l'inizializzazione e il recupero delle informazioni sugli strumenti disponibili. **`eks-mcp:CallReadOnlyTool`**è necessario per l'utilizzo di strumenti di sola lettura ed **`eks-mcp:CallPrivilegedTool`** è richiesto per l'utilizzo di strumenti di accesso completo (scrittura).

## Variabili di ambiente

**`AWS_PROFILE`**(opzionale) nome del profilo delle AWS credenziali da utilizzare; può essere sovrascritto dall'argomento della riga di comando. `--profile`

- Ad esempio: `export AWS_PROFILE=production`

**`AWS_REGION`**(opzionale) AWS regione per la firma SIGv4; il valore predefinito è se non è impostato. `us-west-2`

- Ad esempio: `export AWS_REGION=us-east-1`

### Arguments (Argomenti)

URL dell'endpoint MCP Sigv4 (obbligatorio) L'URL dell'endpoint MCP a cui connettersi.

- Ad esempio: `https://eks-mcp.us-west-2.api.aws/mcp`

**`--service`**(opzionale) nome del AWS servizio per la firma SIGv4; rilevato automaticamente dal nome host dell'endpoint se non fornito.

- Ad esempio: `--service eks-mcp`

**`--profile`**(opzionale) AWS profilo delle credenziali da utilizzare. Il valore predefinito è la variabile di `AWS_PROFILE` ambiente se non specificata.

- Ad esempio: `--profile production`

**`--region`** AWS regione da usare. Utilizza la variabile di `AWS_REGION` ambiente se non è impostata, il valore predefinito è. `us-east-1`

- Ad esempio: `--region us-west-2`

**`--read-only`**(opzionale) Disabilita gli strumenti che potrebbero richiedere autorizzazioni di scrittura (gli strumenti che NON richiedono autorizzazioni di scrittura sono annotati con read =true). OnlyHint Per impostazione predefinita, tutti gli strumenti sono abilitati.

- Ad esempio: `--read-only`

Per ulteriori opzioni di configurazione, vedere Parametri di configurazione.

### Tools (Strumenti)

Per informazioni dettagliate su tutti gli strumenti disponibili, inclusi parametri ed esempi di utilizzo, vederethe section called "Tools (Strumenti)".

## Riferimento agli strumenti del server MCP di Amazon EKS

Il server espone i seguenti strumenti MCP.

> **Note**
>
> Il server MCP Amazon EKS è in versione di anteprima per Amazon EKS ed è soggetto a modifiche.

## Read-only strumenti

Questa sezione descrive gli strumenti di sola lettura disponibili per il server EKS MCP. Nota che tutte le operazioni dell'API Kubernetes in sola lettura possono accedere a entrambe:

- Cluster privati (vedi Cluster private endpoint) https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html#cluster-endpoint-private
- Cluster pubblici

### search_eks_documentation

Cerca nella documentazione EKS informazioni e linee guida aggiornate. Questo strumento fornisce l'accesso alla documentazione EKS più recente, comprese nuove funzionalità e miglioramenti recenti di cui gli agenti potrebbero non essere a conoscenza.

Parametri:

- query (obbligatorio): la tua domanda specifica o la query di ricerca relativa alla documentazione, alle funzionalità o alle best practice di EKS.
- limite (opzionale): numero massimo di risultati della documentazione da restituire (1-10). Impostazione predefinita: 5.

### search_eks_troubleshooting_guide

Cerca nella Guida alla risoluzione dei problemi di EKS informazioni sulla risoluzione dei problemi in base a una query. Aiuta a identificare i problemi più comuni e fornisce soluzioni dettagliate.

Parametri:

- query (obbligatorio): la tua domanda specifica o la descrizione del problema relativa alla risoluzione dei problemi EKS.

describe_eks_resource

Recupera informazioni dettagliate su una specifica risorsa del cluster EKS, tra cui configurazione, stato e metadati.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS (richiesto per le risorse con ambito di cluster).
- resource_type (obbligatorio): il tipo di risorsa EKS da descrivere. Valori validi:
- accessentry (richiede cluster_name e resource_name come PrincipalArn)
- addon (richiede cluster_name e resource_name come nome del componente aggiuntivo)
- cluster (richiede cluster_name), nodegroup (richiede cluster_name e resource_name come nome del nodegroup).
- resource_name (opzionale): nome della risorsa specifica da descrivere (obbligatorio per la maggior parte dei tipi di risorse).

list_eks_resources

Elenca le risorse EKS di un tipo specifico, restituendo un riepilogo di tutte le risorse del tipo specificato che sono accessibili.

Parametri:

- resource_type (obbligatorio): il tipo di risorsa EKS da elencare. Valori validi:
- accessentry (richiede cluster_name)
- addon (richiede cluster_name)
- cluster (non sono richiesti parametri aggiuntivi)
- nodegroup (richiede cluster_name).
- cluster_name (opzionale): nome del cluster EKS (richiesto per le risorse con ambito di cluster).

get_eks_insights

Recupera informazioni e consigli sui cluster EKS per l'ottimizzazione. Fornisce informazioni utili per la sicurezza, le prestazioni e l'ottimizzazione dei costi sulla base delle AWS best practice e dell'analisi dei cluster.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS.
- categoria (opzionale): Categoria opzionale in base alla quale filtrare gli approfondimenti (ad esempio, «MISCONFIGURATION» o «UPGRADE_READINESS»).
- insight_id (opzionale): ID opzionale di un approfondimento specifico per cui ottenere informazioni dettagliate.
- next_token (opzionale): token opzionale per l'impaginazione per ottenere la serie successiva di risultati.

get_eks_vpc_config

Recupera la configurazione VPC per un cluster EKS, incluse sottoreti, tabelle di routing e connettività di rete.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS per cui ottenere la configurazione VPC.
- vpc_id (opzionale): ID del VPC specifico da interrogare (opzionale, utilizzerà il cluster VPC se non specificato).

get_k8s_events

Recupera gli eventi Kubernetes relativi a risorse specifiche per la risoluzione dei problemi e il monitoraggio.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS in cui si trova la risorsa.
- tipo (obbligatorio): tipo di oggetto coinvolto (ad esempio, «Pod», «Deployment», «Service»). Deve corrispondere esattamente al tipo di risorsa.
- name (obbligatorio): nome dell'oggetto coinvolto per il quale ottenere gli eventi.
- namespace (opzionale): Namespace dell'oggetto coinvolto. Richiesto per le risorse con namespace (come Pods, Deployments). Non richiesto per le risorse con ambito di cluster (come Nodes,). PersistentVolumes

get_pod_logs

Recupera i log dai pod in un cluster EKS con opzioni di filtro.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS in cui è in esecuzione il pod.
- namespace (obbligatorio): spazio dei nomi Kubernetes in cui si trova il pod.
- pod_name (obbligatorio): nome del pod da cui recuperare i log.
- container_name (opzionale): nome del contenitore specifico da cui ottenere i log. Richiesto solo se il pod contiene più contenitori.
- limit_bytes (opzionale): numero massimo di byte da restituire. Predefinito: 10 KB (10240 byte).
- precedente (opzionale): restituisce i registri dei contenitori terminati precedenti (il valore predefinito è false). Utile per ottenere i log dei pod che si stanno riavviando.
- since_seconds (opzionale): restituisce solo i log più recenti di questo numero di secondi. Utile per ottenere i log recenti senza recuperare l'intera cronologia.
- tail_lines (opzionale): numero di righe da restituire dalla fine dei log. Predefinito: 100.

list_api_versions

Elenca tutte le versioni API disponibili nel cluster Kubernetes specificato.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS.

list_k8s_resources

Elenca le risorse Kubernetes di un tipo specifico in un cluster EKS.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS in cui si trovano le risorse.
- tipo (obbligatorio): tipo di risorse Kubernetes da elencare (ad esempio, 'Pod', 'Service', 'Deployment'). Usa `list_api_versions` lo strumento per trovare i tipi di risorse disponibili.
- api_version (obbligatorio): versione API delle risorse Kubernetes (ad esempio, 'v1', '', 'networking.k8s. apps/v1 io/v1'). Usa lo `list_api_versions` strumento per trovare le versioni API disponibili.
- field_selector (opzionale): selettore di campo per filtrare le risorse (ad esempio, 'metadata.name=my-pod, status.phase=running'). Usa la stessa sintassi del flag --field-selector di kubectl.

- label_selector (opzionale): selettore di etichette per filtrare le risorse (ad esempio, 'app=nginx, tier=frontend'). Usa la stessa sintassi del flag di `--selector` kubectl.
- namespace (opzionale): Namespace delle risorse Kubernetes da elencare. Se non fornite, le risorse verranno elencate in tutti i namespace (per le risorse con namespace).

read_k8s_resource

Recupera informazioni dettagliate su una specifica risorsa Kubernetes in un cluster EKS.

Parametri:

- api_version (obbligatorio): versione API della risorsa Kubernetes (ad esempio, «v1"," «, «networking.k8s. apps/v1 io/v1«).
- cluster_name (obbligatorio): nome del cluster EKS in cui si trova la risorsa.
- tipo (obbligatorio): tipo di risorsa Kubernetes (ad esempio, «Pod», «Service», «Deployment»).
- nome (obbligatorio): nome della risorsa Kubernetes da leggere.
- namespace (opzionale): Namespace della risorsa Kubernetes. Obbligatorio per le risorse con namespace. Non richiesto per le risorse con ambito di cluster (come Nodes,). PersistentVolumes

generate_app_manifest

Genera manifesti di distribuzione e servizio Kubernetes standardizzati per applicazioni containerizzate.

Parametri:

- app_name (obbligatorio): nome dell'applicazione. Utilizzato per i nomi di distribuzione e di servizio e per le etichette.
- image_uri (obbligatorio): URI completo dell'immagine ECR con tag (ad esempio, 123456789012.dkr.ecr.region.amazonaws. com/repoimage_uri (obbligatorio): URI completo dell'immagine ECR con tag (ad esempio, 123456789012.dkr.ecr.region.amazonaws. ----sep----:tag). Deve includere il percorso e il tag completi del repository.
- load_balancer_scheme (opzionale): schema di bilanciamento del carico. AWS Valori validi:
- «interno» (solo VPC privato)
- «rivolto a Internet» (accesso pubblico).
- Predefinito: «interno».

- cpu (opzionale): richiesta di CPU per ogni contenitore (ad esempio, «100m» per 0,1 core CPU, «500m» per mezzo core). Predefinito: «100m».
- memoria (opzionale): richiesta di memoria per ogni contenitore (ad esempio, «128Mi» per 128 MiB, «1Gi» per 1 GiB). Predefinito: «128Mi»
- namespace (opzionale): namespace Kubernetes in cui distribuire l'applicazione. Predefinito: «predefinito».
- porta (opzionale): porta del contenitore su cui l'applicazione è in ascolto. Impostazione predefinita: 80
- repliche (opzionale): numero di repliche da distribuire. Impostazione predefinita: 2

get_cloudwatch_logs

Interroga i CloudWatch log con filtraggio basato sui parametri di input e supporto per i gruppi di log standard utilizzati per l'osservabilità dei cluster EKS.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS in cui si trova la risorsa. Utilizzato per costruire il nome del gruppo di log CloudWatch .
- resource_type (obbligatorio): tipo di risorsa da cercare nei log. Valori validi: +"pod», «node», «container», «cluster». Questo determina il modo in cui i log vengono filtrati.
- log_type (obbligatorio): tipo di log da interrogare. Valori validi:
- «applicazione»: registri Container/application
- «host»: registri Node-level di sistema
- «performance»: registri delle metriche delle prestazioni
- «piano di controllo»: registri del piano di controllo EKS
- «your-log-group-name»: fornisci direttamente un nome di gruppo di log personalizzato. CloudWatch
- resource_name (opzionale): nome della risorsa da cercare nei messaggi di log (ad esempio, nome del pod, nome del nodo, nome del contenitore). Utilizzato per filtrare i log per la risorsa specifica.
- minuti (opzionale): numero di minuti per la ricerca dei log. Predefinito: 15. Ignorato se viene fornito start_time. Utilizza valori più bassi per i numeri recenti, valori più grandi per l'analisi storica.
- start_time (opzionale): ora di inizio in formato ISO (ad esempio, «T00:00:00Z2023-01-01 «). Se fornito, sovrascrive il parametro dei minuti.

- end_time (opzionale): ora di fine in formato ISO (ad esempio, «2023-01-01 «). T01:00:00Z Se non viene fornito, il valore predefinito è l'ora corrente.
- campi (opzionale): campi personalizzati da includere nei risultati della query (il valore predefinito è "@timestamp, @message «). Utilizza la sintassi CloudWatch dei campi Logs Insights.
- filter_pattern (opzionale): pattern di filtro CloudWatch Logs aggiuntivo da applicare. Utilizza la sintassi CloudWatch Logs Insights (ad esempio, «ERROR», «field=value»).
- limite (opzionale): numero massimo di voci di registro da restituire. Utilizza valori più bassi (10-50) per interrogazioni più veloci, valori più alti (100-1000) per risultati più completi. I valori più alti possono inficiare le prestazioni.

get_cloudwatch_metrics

Recupera CloudWatch metriche e punti dati per il monitoraggio dei cluster EKS e l'analisi delle prestazioni. Gestisce le metriche di Container Insights, le metriche personalizzate e i periodi di tempo e le dimensioni configurabili.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS per cui ottenere le metriche.
- dimensioni (obbligatorio): dimensioni da utilizzare per l'interrogazione delle CloudWatch metriche come stringa JSON. Deve includere dimensioni appropriate per il tipo di risorsa e la metrica (ad esempio, '{» ClusterName «: «my-cluster», "«: PodName «my-pod», «Namespace»: «default"}').
- metric_name (obbligatorio): nome della metrica da recuperare. Esempi comuni:
- `cpu_usage_total`: utilizzo totale della CPU
- `memory_rss`: utilizzo della memoria Resident Set Size
- `network_rx_bytes`: byte di rete ricevuti
- `network_tx_bytes`: byte di rete trasmessi
- namespace (obbligatorio): CloudWatch namespace in cui è memorizzata la metrica. Valori comuni:
- `"ContainerInsights"`: per le metriche dei contenitori
- `"AWS/EC2"`: Per le metriche delle istanze EC2
- `"AWS/EKS"`: Per le metriche del piano di controllo EKS
- minuti (opzionale): numero di minuti per la ricerca delle metriche. Predefinito: 15. Ignorato se viene fornito start_time.

- start_time (opzionale): ora di inizio in formato ISO (ad esempio, «2023-01-01 «). T00:00:00Z Se fornito, sovrascrive il parametro dei minuti.
- end_time (opzionale): ora di fine in formato ISO (ad esempio, «2023-01-01 «). T01:00:00Z Se non viene fornito, il valore predefinito è l'ora corrente.
- limite (opzionale): numero massimo di punti dati da restituire. Valori più alti (100-1000) forniscono dati più granulari ma possono influire sulle prestazioni. Impostazione predefinita: 50.
- periodo (opzionale): periodo in secondi per i punti dati metrici. Predefinito: 60 (1 minuto). Valori più bassi (1-60) forniscono una risoluzione più elevata ma potrebbero essere meno disponibili.
- stat (opzionale): statistica da utilizzare per l'aggregazione delle metriche. Predefinito: «Average». Valori validi:
- `Average`: Valore medio durante il periodo
- `Sum`: Valore totale nel periodo
- `Maximum`: Valore massimo durante il periodo
- `Minimum`: valore più basso durante il periodo
- `SampleCount`: Numero di campioni durante il periodo.

get_eks_metrics_guidance

Ottieni indicazioni sulle CloudWatch metriche per tipi di risorse specifici nei cluster EKS.
Utile per l'agente per determinare le dimensioni corrette da utilizzare con lo strumento get_cloudwatch_metrics.

Parametri:

- resource_type (obbligatorio): tipo di risorsa per cui ottenere le metriche (cluster, nodo, pod, namespace, servizio).

get_policies_for_role

Recupera tutte le policy associate a un ruolo IAM specificato, incluse le policy di assunzione del ruolo, le policy gestite e le policy inline.

Parametri:

- role_name (obbligatorio): nome del ruolo IAM per cui ottenere le policy. Il ruolo deve esistere nel tuo account. AWS

## Strumenti di accesso completo (scrittura)

Questa sezione descrive gli strumenti di accesso completo (scrittura) disponibili per il server EKS MCP. Nota che (a partire da «oggi») tutte le operazioni di scrittura dell'API Kubernetes possono accedere solo a:

- Cluster pubblici (endpoint =true) PublicAccess

### manage_k8s_resource

Gestisce una singola risorsa Kubernetes con operazioni di scrittura (creazione, aggiornamento, patch o eliminazione).

Parametri:

- operazione (obbligatorio): operazione da eseguire sulla risorsa. Valori validi:
- `create:`Crea una nuova risorsa
- `replace`: Sostituisce una risorsa esistente
- `patch`: aggiorna campi specifici di una risorsa esistente
- `delete`: Elimina una risorsa esistente
- Nota: usa read_k8s_resource per leggere le risorse e list_k8s_resources per elencare più risorse.
- cluster_name (obbligatorio): nome del cluster EKS in cui si trova o verrà creata la risorsa.
- tipo (obbligatorio): tipo di risorsa Kubernetes (ad esempio, «Pod», «Service», «Deployment»).
- api_version (obbligatorio): versione API della risorsa Kubernetes (ad es. «v1"," «, «networking.k8s. apps/v1 io/v1«).
- body (opzionale): definizione della risorsa come stringa JSON. Necessario per le operazioni di creazione, sostituzione e patch. Per creare e sostituire, questa dovrebbe essere una definizione completa della risorsa. Per la patch, dovrebbe contenere solo i campi da aggiornare.
- name (opzionale): nome della risorsa Kubernetes. Richiesto per tutte le operazioni tranne create (dove può essere specificato nel corpo).
- namespace (opzionale): Namespace della risorsa Kubernetes. Obbligatorio per le risorse con namespace. Non richiesto per le risorse con ambito di cluster (come Nodes,). PersistentVolumes

### apply_yaml

Applica i manifesti YAML di Kubernetes a un cluster EKS.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS in cui verranno create o aggiornate le risorse.
- namespace (obbligatorio): namespace Kubernetes a cui applicare le risorse. Verrà utilizzato per le risorse con namespace che non specificano un namespace.
- yaml_content (obbligatorio): contenuto YAML da applicare al cluster. Può contenere più documenti separati da '---'.
- force (opzionale): se aggiornare le risorse se già esistono (simile a kubectl apply). Impostato su false per creare solo nuove risorse.

manage_eks_stacks

Gestisce gli CloudFormation stack EKS con operazioni per la generazione di modelli, l'implementazione, la descrizione e l'eliminazione dei cluster EKS e della relativa infrastruttura sottostante. La creazione del cluster richiede in genere 15-20 minuti per essere completata. Per le operazioni di distribuzione ed eliminazione, lo stack deve essere stato creato da questo strumento (ad esempio, contrassegnato con CreatedBy =EksMcpServer).

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS (per le operazioni di generazione, distribuzione, descrizione ed eliminazione). Questo nome verrà utilizzato per derivare il nome dello CloudFormation stack e verrà incorporato nelle risorse del cluster.
- operazione (obbligatorio): operazione da eseguire. Valori validi:
- `generate`: Genera un CloudFormation modello
- `deploy`: Distribuisci uno CloudFormation stack (richiede`template_content`)
- `describe`: Describe/read uno CloudFormation stack (di sola lettura)
- `delete`: Elimina una pila CloudFormation
- template_content (opzionale): contenuto del CloudFormation modello (per le operazioni di distribuzione). Dovrebbe essere il contenuto completo del modello YAML o JSON. Supporta sia risorse singole che contenuti YAML composti da più documenti separati da «---».

add_inline_policy

Aggiunge una nuova policy in linea a un ruolo IAM.

Parametri:

- autorizzazioni (obbligatorie): autorizzazioni da includere nella policy come stringhe JSON che rappresentano le dichiarazioni della politica IAM. Può essere una singola stringa JSON o un array di stringhe JSON.
- policy_name (obbligatorio): nome della policy in linea da creare. Deve essere univoco all'interno del ruolo.
- role_name (obbligatorio): nome del ruolo IAM a cui aggiungere la policy. Il ruolo deve esistere.

# Usa Amazon Q Developer sulla console Amazon EKS

Amazon Elastic Kubernetes Service (EKS) si integra con Amazon Q per fornire la AI-powered risoluzione dei problemi direttamente nella console di gestione. AWS Questa integrazione consente di analizzare e risolvere rapidamente i problemi relativi a cluster, piano di controllo, nodo e carico di lavoro con l'assistenza AI di Amazon Q.

## Come funziona

La console Amazon EKS visualizza contestualmente i pulsanti Inspect with Amazon Q insieme a errori o problemi in tutta la console. Facendo clic su questo pulsante, Amazon Q analizza automaticamente il problema e apre un pannello di chat sul lato destro della console con i risultati delle indagini, l'analisi delle cause principali e le procedure di mitigazione suggerite.

L'integrazione viene visualizzata nelle seguenti posizioni all'interno della console Amazon EKS:

- Integrità del cluster: analizza i problemi di salute del cluster e i messaggi di stato nella scheda Integrità del cluster nella dashboard di osservabilità
- Piano di controllo: risolvi gli errori e gli avvisi dei componenti del piano di controllo nel monitoraggio del piano di controllo nella dashboard di osservabilità
- Informazioni sull'aggiornamento: analizza i potenziali ostacoli agli aggiornamenti e i problemi di compatibilità nella dashboard di osservabilità di Upgrade Insights
- Stato del nodo: analizza i problemi a livello di nodo che influiscono sulla capacità del cluster nei problemi di salute dei nodi nella dashboard di osservabilità
- Carichi di lavoro: analizza gli eventi Kubernetes sui pod che indicano guasti o problemi

## Utilizzo di Amazon Q per la risoluzione dei problemi

Per indagare su un problema con Amazon Q

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster da analizzare.
3. Quando trovi un messaggio di errore o un indicatore di problema, cerca il pulsante Ispeziona con Amazon Q. Il pulsante viene visualizzato contestualmente accanto al problema o nella visualizzazione dei dettagli sullo stato dell'errore.
4. Scegli Ispeziona con Amazon Q.
5. Amazon Q analizza automaticamente il problema e visualizza l'analisi in un pannello di chat sul lato destro della console.
6. Esamina i risultati dell'indagine, inclusa l'analisi della causa principale e le misure di mitigazione suggerite.
7. Puoi continuare la conversazione ponendo domande di follow-up ad Amazon Q sul problema.

Nota: l'integrazione di Amazon Q viene visualizzata solo quando si verifica un errore, un avviso o un problema che richiede un'indagine. Non viene visualizzato quando le risorse sono intatte.

## Considerazioni

Quando usi Amazon Q con Amazon EKS, considera quanto segue:

- Read-only operazioni: l'integrazione di Amazon Q esegue solo operazioni di lettura sulle risorse del cluster. Non apporta alcuna modifica o azione di scrittura alla configurazione del cluster o ai carichi di lavoro.
- Cross-region elaborazione: Amazon Q può elaborare dati in diverse AWS regioni per fornire AI-powered analisi. Per ulteriori informazioni sull'elaborazione tra regioni, consulta l'Cross-region elaborazione nella Amazon Q Developer User Guide.
- AWS Solo console di gestione: questa integrazione è disponibile solo tramite la console di AWS gestione. Non è disponibile tramite la AWS CLI, le AWS API o l'infrastruttura come strumenti di codice.

## Ulteriori informazioni

Per ulteriori informazioni sull'utilizzo di Amazon Q, consulta Chat with Amazon Q nella Amazon Q Developer User Guide.

# Controllo delle versioni su Amazon EKS

Questa sezione è progettata per fornire un supporto nell'ambito del funzionamento del controllo delle versioni Kubernetes all'interno di Elastic Kubernetes Service (EKS) di Amazon. Sono riportati i dettagli sulle versioni supportate, sulle nostre policy obsolete e di supporto, sui processi di aggiornamento e sulle considerazioni chiave per la gestione delle versioni del cluster.

Argomenti


- Comprendere il ciclo di vita delle versioni di Kubernetes su EKS
- Visualizzazione delle versioni della piattaforma Amazon EKS per ciascuna versione Kubernetes


## Comprendere il ciclo di vita delle versioni di Kubernetes su EKS

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS.

Kubernetes è in rapida evoluzione, con nuovi aggiornamenti di progettazione, funzionalità e correzioni di bug. La community rilascia nuove versioni di Kubernetes secondarie (ad esempio, `1.36`) in media una volta ogni quattro mesi. Amazon EKS segue il ciclo di rilascio e deprecazione originario per le versioni secondarie. Man mano che nuove versioni di Kubernetes diventano disponibili in Amazon EKS, si consiglia di aggiornare tempestivamente i cluster in modo da usare la versione più recente disponibile.

Una versione secondaria è supportata come standard in Amazon EKS per i primi 14 mesi dopo il rilascio. Una volta superata la data di fine del supporto standard, la versione passa al supporto esteso per i 12 mesi successivi. Il supporto esteso consente di mantenere una versione Kubernetes specifica di più a lungo a un costo aggiuntivo per ora del cluster. Se non hai aggiornato il cluster prima della fine del periodo di supporto esteso, il cluster viene aggiornato automaticamente alla versione estesa più vecchia attualmente supportata.

Il supporto esteso è abilitato per impostazione predefinita. Per disabilitarlo, consulta Disable EKS extended support.

Ti consigliamo di creare il tuo cluster con l'ultima versione di Kubernetes disponibile supportata da Amazon EKS. Se la propria applicazione richiede una versione specifica di Kubernetes, è possibile

selezionare versioni precedenti. Puoi creare nuovi cluster Amazon EKS su qualsiasi versione offerta con supporto standard o esteso.

## Versioni disponibili con supporto standard

Le seguenti versioni di Kubernetes sono attualmente disponibili nel supporto standard di Amazon EKS:

- `1.36`
- `1.35`
- `1.34`

Per le modifiche importanti di cui tenere conto per ogni versione del supporto standard, consulta Kubernetes versions standard support.

## Versioni disponibili con supporto esteso

Le seguenti versioni di Kubernetes sono attualmente disponibili nel supporto esteso di Amazon EKS:

- `1.33`

- `1.32`
- `1.31`

Per le modifiche importanti di cui tenere conto per ogni versione del supporto esteso, consulta Kubernetes versions extended support.

## Rollback della versione

I cluster aggiornati tramite l'aggiornamento sul posto possono essere ripristinati alla versione secondaria precedente entro 7 giorni dal completamento dell'aggiornamento. Per tornare a una versione con supporto esteso, devi prima modificare la politica di aggiornamento del cluster in. `EXTENDED` I cluster che sono stati aggiornati automaticamente al termine del supporto esteso non possono essere ripristinati. Per ulteriori informazioni, consulta the section called "Ripristina la versione di Kubernetes".

## Calendario di rilascio di Amazon EKS Kubernetes

La tabella seguente mostra le date importanti di rilascio e supporto da considerare per ciascuna versione di Kubernetes. La fatturazione per il supporto esteso parte all'inizio del giorno in cui la versione raggiunge la fine del supporto standard, nel fuso orario UTC+0. Le date nella tabella seguente utilizzano il fuso orario UTC+0.

> **Note**
>
> Le date con solo un mese e un anno sono approssimative e vengono aggiornate con una data esatta quando nota.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/versioning/kubernetes-versions.adoc.atom
```

| Versione di Kubernetes | Versione upstream | Rilascio Amazon EKS | Data di fine del supporto standard | Data di fine del supporto esteso |
|---|---|---|---|---|
| 1.36 | 22 aprile 2026 | 2 giugno 2026 | 2 agosto 2027 | 2 agosto 2028 |
| 1.35 | 17 dicembre 2025 | 27 gennaio 2026 | 27 marzo 2027 | 27 marzo 2028 |
| 1.34 | 27 agosto 2025 | 2 ottobre 2025 | 2 dicembre 2026 | 2 dicembre 2027 |
| 1.33 | 23 aprile 2025 | 29 maggio 2025 | 29 luglio 2026 | 29 luglio 2027 |
| 1.32 | 11 dicembre 2024 | 23 gennaio 2025 | 23 marzo 2026 | 23 marzo 2027 |
| 1.31 | 13 agosto 2024 | 26 settembre 2024 | 26 novembre 2025 | 26 novembre 2026 |

## Ottieni informazioni sulla versione con AWS CLI

Puoi utilizzare la AWS CLI per ottenere informazioni sulle versioni di Kubernetes disponibili su EKS, ad esempio la data di fine del supporto standard.

### Per recuperare informazioni sulle versioni di Kubernetes disponibili su EKS utilizzando AWS CLI

1. Apri il terminale.
2. Assicurati di avere la AWS CLI installata e configurata. Per ulteriori informazioni, vedi Installing or updating to the latest version of the CLI.
3. Esegui il comando seguente:

```
aws eks describe-cluster-versions
```

4. Il comando restituirà un output JSON con dettagli sulle versioni del cluster disponibili. Ecco un esempio dell'output:

```
{
```

```
    "clusterVersions": [
        {
            "clusterVersion": "1.31",
            "clusterType": "eks",
            "defaultPlatformVersion": "eks.21",
            "defaultVersion": true,
            "releaseDate": "2024-09-25T17:00:00-07:00",
            "endOfStandardSupportDate": "2025-11-25T16:00:00-08:00",
            "endOfExtendedSupportDate": "2026-11-25T16:00:00-08:00",
            "status": "STANDARD_SUPPORT",
            "kubernetesPatchVersion": "1.31.3"
        }
    ]
}
```

L'output fornisce le seguenti informazioni per ogni versione del cluster:

- `clusterVersion`: la versione Kubernetes del cluster EKS
- `clusterType`: il tipo di cluster (ad esempio, "eks")
- `defaultPlatformVersion`: la versione predefinita della piattaforma EKS
- `defaultVersion`: se si tratta della versione predefinita
- `releaseDate`: la data in cui è stata rilasciata questa versione
- `endOfStandardSupportDate`: la data di fine del supporto standard
- `endOfExtendedSupportDate`: la data di fine del supporto esteso
- `status`: lo stato del supporto corrente della versione, ad esempio `STANDARD_SUPPORT` o `EXTENDED_SUPPORT`
- `kubernetesPatchVersion`: la versione specifica della patch Kubernetes

## Domande frequenti sulle versioni di Amazon EKS

Quante versioni di Kubernetes sono disponibili nel supporto standard?

> Amazon EKS, in linea con il supporto della community Kubernetes per le versioni di Kubernetes, si impegna a offrire il supporto per almeno tre versioni di Kubernetes in qualunque momento. L'annuncio della data di fine supporto standard di una qualsiasi versione di Kubernetes secondaria sarà comunicato almeno 60 giorni prima. A causa del processo di qualifica e rilascio

di Amazon EKS per le nuove versioni di Kubernetes, la data di fine supporto standard di una versione di Kubernetes in Amazon EKS avverrà in corrispondenza o dopo la data in cui il progetto Kubernetes smetterà di supportare la versione a monte.

Per quanto tempo una versione di Kubernetes riceve il supporto standard da Amazon EKS?

Una versione di Kubernetes riceve supporto standard per 14 mesi dopo la prima disponibilità su Amazon EKS. Questo è vero anche se Kubernetes upstream non supporta più una versione disponibile su Amazon EKS. Viene eseguito il backporting delle patch di sicurezza applicabili alle versioni di Kubernetes che sono supportate in Amazon EKS.

Riceverò un avviso quando il supporto standard per una versione Kubernetes su Amazon EKS è terminato?

Sì. Se su un cluster del tuo account è in esecuzione la versione prossima alla fine del supporto, Amazon EKS invia un avviso tramite AWS Health Dashboard circa 12 mesi dopo il rilascio della versione Kubernetes su Amazon EKS. L'avviso include la data di fine supporto, che è successiva di almeno 60 giorni alla data dell'invio dell'avviso.

Quali funzionalità di Kubernetes sono supportate da Amazon EKS?

Amazon EKS supporta tutte le funzionalità di disponibilità generale (GA) dell'API di Kubernetes. Per impostazione predefinita, le nuove API beta non sono abilitate nei cluster. Tuttavia, per impostazione predefinita, le API beta precedentemente esistenti e le nuove versioni delle API beta esistenti continuano a essere abilitate. Le funzionalità alfa non sono supportate.

I gruppi di nodi gestiti da Amazon EKS vengono aggiornati automaticamente insieme alla versione del piano di controllo del cluster?

No, un gruppo di nodi gestiti crea istanze Amazon EC2 nel tuo account. Queste istanze non vengono aggiornate in automatico quando l'utente o Amazon EKS aggiorna il piano di controllo. Per ulteriori informazioni, consulta the section called "Aggiorna". Si consiglia di mantenere la stessa versione di Kubernetes sul piano di controllo e sui nodi.

I gruppi di nodi autogestiti vengono aggiornati automaticamente insieme alla versione del piano di controllo del cluster?

No, un gruppo di nodi autogestito include le istanze di Amazon EC2 nel tuo account. Queste istanze non vengono aggiornate in automatico quando tu o Amazon EKS aggiornate la versione del piano di controllo per tuo conto. Nella console, un gruppo di nodi autogestito non riceve alcuna indicazione di aggiornamento. È possibile visualizzare la versione di `kubelet` installata su un nodo selezionando il nodo nella finestra di dialogo Nodi nella scheda Panoramica del cluster per

determinare quali nodi devono essere aggiornati. È necessario aggiornare manualmente i nodi. Per ulteriori informazioni, consulta the section called "Metodi di aggiornamento".

Il progetto Kubernetes verifica la compatibilità tra il piano di controllo e i nodi per un massimo di due versioni secondarie. Ad esempio, i nodi `1.33` continuano a funzionare se orchestrati da un piano di controllo `1.36`. Tuttavia, non è consigliabile l'esecuzione di un cluster con nodi aggiornati in modo persistente a tre versioni secondarie precedenti rispetto al piano di controllo. Per ulteriori informazioni, consulta Policy per la versione Kubernetes e il supporto Skew della versione nella documentazione di Kubernetes. Si consiglia di mantenere la stessa versione di Kubernetes sul piano di controllo e sui nodi.

I pod in esecuzione su Fargate vengono aggiornati automaticamente con un aggiornamento automatico della versione del piano di controllo cluster?

No. Consigliamo di eseguire i pod Fargate come parte di un controller di replica, così come avviene in un'implementazione di Kubernetes. Quindi esegui un riavvio continuo di tutti i pod Fargate. La nuova versione del pod Fargate viene implementata con una versione `kubelet` che corrisponde alla versione aggiornata del piano di controllo del cluster. Per ulteriori informazioni, consulta l'argomento relativo alla funzionalità Deployments nella documentazione Kubernetes.

> **Important**
>
> Se aggiorni il piano di controllo, dovrai comunque aggiornare personalmente i nodi Fargate. Per aggiornare i nodi Fargate, elimina il pod Fargate rappresentato dal nodo e implementalo nuovamente. Il nuovo pod viene implementato con una versione `kubelet` che corrisponde alla versione del cluster.

Quali versioni di Kubernetes sono supportate per i nodi ibridi?

Amazon EKS Hybrid Nodes supporta le stesse versioni di Kubernetes dei cluster Amazon EKS con altri tipi di calcolo dei nodi, incluso il supporto della versione Kubernetes standard ed esteso. I nodi ibridi non vengono aggiornati automaticamente quando si aggiorna la versione del piano di controllo, perché ne è responsabile l'utente. Per ulteriori informazioni, consulta the section called "Aggiornamento dei nodi ibridi".

# Amazon EKS extended support FAQs

La terminologia del supporto standard e del supporto esteso è una novità per me. Cosa significano questi termini?

Il supporto standard per una versione di Kubernetes in Amazon EKS inizia quando viene rilasciata una versione Kubernetes su Amazon EKS e termina 14 mesi dopo la data di rilascio. Il supporto esteso per una versione di Kubernetes inizierà immediatamente dopo la fine del supporto standard e terminerà trascorsi i 12 mesi successivi. Ad esempio, il supporto standard per la versione `1.23` in Amazon EKS termina l'11 ottobre 2023. Il supporto esteso per la versione `1.23` è iniziato il 12 ottobre 2023 ed è terminato l'11 ottobre 2024.

Cosa devo fare per ottenere il supporto esteso per i cluster Amazon EKS?

Dovrai abilitare il supporto esteso (vedi EKS extended support) per il tuo cluster modificando la policy di aggiornamento del cluster in EXTENDED. Per impostazione predefinita su tutti i cluster nuovi ed esistenti, la policy di aggiornamento è impostata su EXTENDED, se non diversamente specificato. Consulta Cluster upgrade policy per visualizzare la policy di aggiornamento del cluster. Il supporto standard inizia quando viene rilasciata una versione di Kubernetes su Amazon EKS e termina 14 mesi dopo la data di rilascio. Il supporto esteso per una versione di Kubernetes inizierà immediatamente dopo la fine del supporto standard e terminerà trascorsi i 12 mesi successivi.

Per quali versioni Kubernetes posso ottenere il supporto esteso?

Puoi eseguire i cluster su qualsiasi versione per un massimo di 12 mesi dopo la fine del supporto standard per quella versione. Ciò significa che ogni versione sarà supportata per 26 mesi in Amazon EKS (14 mesi di supporto standard più 12 mesi di supporto esteso).

Cosa succede se non voglio utilizzare l'assistenza estesa?

Se non desideri ricevere automaticamente il supporto esteso, puoi aggiornare il cluster a una versione di Kubernetes che includa il supporto standard di Amazon EKS. Per disabilitare il supporto esteso, consulta Disable EKS extended support. Nota: se disabiliti il supporto esteso, il tuo cluster sarà automaticamente aggiornato al termine del supporto standard.

Cosa succederà alla fine dei 12 mesi di supporto esteso?

I cluster in esecuzione su una versione di Kubernetes che ha completato il ciclo di vita di 26 mesi (14 mesi di supporto standard più 12 mesi di supporto esteso) verranno aggiornati automaticamente alla versione successiva. L'aggiornamento automatico include solo il piano di controllo Kubernetes. Se si dispone di nodi EKS Auto Mode, è possibile che si aggiornino

automaticamente. Self-managed i nodi e i gruppi di nodi gestiti da EKS rimarranno nella versione precedente.

Allo scadere della data di fine di supporto, non è più possibile creare nuovi cluster Amazon EKS con la versione non supportata. I piani di controllo esistenti vengono aggiornati in automatico da Amazon EKS alla prima versione supportata attraverso un processo di implementazione graduale dopo la data di fine supporto. Dopo l'aggiornamento automatico del piano di controllo, è necessario aggiornare manualmente i componenti aggiuntivi del cluster e i nodi Amazon EC2. Per ulteriori informazioni, consulta the section called "Aggiornamento della versione di Kubernetes".

Allo scadere della data di fine supporto esteso, quando avverrà esattamente l'aggiornamento automatico del piano di controllo?

Amazon EKS non è in grado di fornire periodi di tempo specifici. Gli aggiornamenti automatici possono avvenire in qualsiasi momento dopo la data di fine supporto esteso. Non riceverai alcuna notifica prima dell'aggiornamento. Consigliamo di aggiornare in modo proattivo il piano di controllo senza fare affidamento sul processo di aggiornamento automatico di Amazon EKS. Per ulteriori informazioni, consulta the section called "Aggiornamento della versione di Kubernetes".

Posso lasciare il mio piano di controllo su una versione Kubernetes a tempo indeterminato?

No. La sicurezza del cloud AWS è la massima priorità. Dopo un certo periodo (di solito 1 anno), la community di Kubernetes interrompe il rilascio di patch per vulnerabilità ed esposizioni (CVE) comuni e sconsiglia il loro invio per le versioni non supportate. Ciò significa che le vulnerabilità specifiche di una versione precedente di Kubernetes potrebbero non essere segnalate, lasciando i cluster esposti senza preavviso e senza opzioni di correzione in caso di vulnerabilità. Pertanto, Amazon EKS non consente ai piani di controllo di rimanere a una versione non più coperta dal supporto esteso.

Sono previsti costi aggiuntivi per ottenere un supporto esteso?

Sì, sono previsti costi aggiuntivi per i cluster Amazon EKS in esecuzione con il supporto esteso. Per informazioni dettagliate sui prezzi, consulta il supporto esteso di Amazon EKS per la versione Kubernetes nel AWS blog o nella nostra pagina dei prezzi.

Cosa è incluso nel supporto esteso?

I cluster Amazon EKS in supporto esteso ricevono patch di sicurezza continue per il piano di controllo di Kubernetes. Inoltre, Amazon EKS rilascerà patch per Amazon VPC CNI, `kube-proxy` e componenti aggiuntivi per le versioni di supporto esteso su CoreDNS. Amazon EKS rilascerà anche patch per le AMI ottimizzate Amazon EKS AWS pubblicate per Amazon Linux,

Bottlerocket e Windows, nonché i nodi Amazon EKS Fargate per tali versioni. Tutti i cluster in Extended Support continueranno a ricevere l'accesso al supporto tecnico da. AWS

Esistono limitazioni alle patch per i componenti non Kubernetes inclusi nel supporto esteso?

Sebbene il supporto esteso copra tutti i componenti specifici di Kubernetes da AWS, fornirà sempre supporto solo per le AMI ottimizzate di Amazon EKS AWS pubblicate per Amazon Linux, Bottlerocket e Windows. Ciò significa che potresti avere componenti più recenti (come sistema operativo o kernel) sulla tua AMI ottimizzata per Amazon EKS durante l'utilizzo del supporto esteso. Ad esempio, quando Amazon Linux 2 raggiungerà la fine del suo ciclo di vita, nel 2025, le AMI Amazon Linux ottimizzate per Amazon EKS verranno create utilizzando un sistema operativo Amazon Linux più recente. Amazon EKS annuncerà e documenterà importanti discrepanze nel ciclo di vita del supporto, come questa, per ogni versione di Kubernetes.

Posso creare nuovi cluster utilizzando una versione con supporto esteso?

Sì.

## Dai un'occhiata alle note di rilascio per le versioni Kubernetes sul supporto standard

> **Tip**
>
> Registrati per i prossimi workshop Amazon EKS.

Questo argomento fornisce importanti modifiche di cui tenere conto per ogni versione Kubernetes del supporto standard. Durante l'aggiornamento, esamina attentamente le modifiche apportate tra la vecchia e la nuova versione del cluster.

### Kubernetes 1.36

Kubernetes `1.36` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.36`, consulta l'official release announcement.

**Important**

- Rimozione del volume GitRepo: il tipo di volume è permanentemente disabilitato in `gitRepo` Kubernetes 1.36 e non può essere riattivato. L'API Kubernetes accetta ancora pod con `gitRepo` volumi, ma kubelet si rifiuterà di eseguirli e restituirà un errore.
  - Azione richiesta: esegui la migrazione ai contenitori init o ai contenitori sidecar git-sync prima di eseguire l'aggiornamento alla 1.36. Per ulteriori informazioni, consulta KEP-5040.
- SELinux Volume Labeling Changes (GA): l'etichettatura dei volumi SELinux più veloce ora viene impostata per impostazione predefinita su tutti i volumi in Kubernetes 1.36, utilizzando invece la rietichettatura ricorsiva dei file. `mount -o context` La condivisione di un volume tra Pod con privilegi e non privilegiati sullo stesso nodo può causare problemi. Le future versioni di Kubernetes potrebbero introdurre ulteriori modifiche sostanziali relative a questa funzionalità.
  - Azione richiesta: i clienti che utilizzano SELinux-enforcing sistemi devono controllare i cluster e assicurarsi che le etichette dei `seLinuxChangePolicy` campi e dei volumi SELinux siano impostate correttamente sui Pod prima dell'aggiornamento. Per maggiori informazioni, consulta SELinux Volume Label Changes goes GA (e probabili implicazioni nella versione 1.37).
- IP/CIDR Validazione rigorosa abilitata per impostazione predefinita: il `StrictIPCIDRValidation` feature gate è ora abilitato per impostazione predefinita per i tipi di API integrati. I campi API non accettano più valori IP o CIDR con zeri iniziali estranei (ad esempio, `010.000.000.005` invece di`10.0.0.5`) o valori CIDR con semantica ambigua (ad esempio, invece di). `192.168.0.5/24` `192.168.0.0/24` Gli oggetti archiviati esistenti vengono conservati tramite il ratcheting di convalida, ma le nuove creazioni e gli aggiornamenti verranno rifiutati. Ciò non si applica ai tipi di risorse personalizzati.
  - Azione richiesta: controlla i manifesti, i grafici Helm e l'automazione per individuare gli indirizzi IP contenenti zeri iniziali o notazione CIDR non canonica. Aggiornali per utilizzare i formati canonici prima dell'aggiornamento. Per ulteriori informazioni, consulta KEP-4858.

- User Namespaces (stabile): User Namespaces fornisce una difesa approfondita associando l'utente root di un container a un utente non privilegiato sull'host, assicurando che un breakout del container non conceda alcun potere amministrativo sul nodo.
  - Per ulteriori informazioni, consulta User Namespaces in Kubernetes are finally GA nel blog di Kubernetes. https://kubernetes.io/blog/2026/04/23/kubernetes-v1-36-userns-ga/
- Resource Health Status (Beta): riporta lo stato di salute di ogni dispositivo nello stato Pod, consentendo agli operatori di determinare se un crash loop è dovuto a uno stato del dispositivo non integro o sconosciuto piuttosto che a problemi dell'applicazione. Funziona sia con i plug-in dei dispositivi che con l'allocazione dinamica delle risorse.
  - Per ulteriori informazioni, consulta KEP-4680.
- Funzionalità di allocazione dinamica delle risorse (beta): diverse funzionalità DRA sono ora abilitate per impostazione predefinita: dispositivi partizionabili e capacità consumabile per una condivisione più granulare di hardware come le GPU e Device Binding Conditions per controlli approfonditi della disponibilità dei dispositivi prima della pianificazione.
  - Per ulteriori informazioni, consulta Kubernetes v1.36: More Drivers, New Features e the Next Era of DRA nel blog di Kubernetes.
- Avviso di deprecazione — Service ExternalIPS: il campo Service è obsoleto in Kubernetes 1.36. `externalIPs .spec` Visualizzerai avvisi di deprecazione durante la creazione o l'aggiornamento dei servizi che utilizzano questo campo. La rimozione completa è prevista per Kubernetes 1.43. I clienti che lo utilizzano `externalIPs` devono migrare a LoadBalancer Services o Gateway API. NodePort https://gateway-api.sigs.k8s.io/ Per ulteriori informazioni, consulta KEP-5707.

Per il changelog completo di Kubernetes`1.36`, consulta https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.36.md

## Kubernetes 1.35

Kubernetes `1.35` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.35`, consulta l'official release announcement.

> **Important**
>
> - Supporto per Cgroup v1 rimosso: Kubernetes 1.35 depreca il supporto per cgroup v1, il che significa che kubelet si rifiuterà di avviarsi per impostazione predefinita sui nodi che utilizzano cgroup v1.

  - AL2023: AL2023 utilizza cgroup v2 per impostazione predefinita e si allinea al comportamento upstream di Kubernetes.
    - Azione richiesta: i clienti che hanno configurato manualmente AL2023 per utilizzare cgroup v1 devono migrare a cgroups v2 o impostarlo manualmente nella configurazione kubelet. https://kubernetes.io/docs/concepts/architecture/cgroups/ `failCgroupV1: false`
  - Bottlerocket: Bottlerocket 1.35 utilizza cgroup v2 per impostazione predefinita, ma viene impostato nella configurazione kubelet, mantenendo la compatibilità con le versioni precedenti. `failCgroupV1: false`
  - Fargate: Fargate continua a utilizzare cgroup v1.
- Fine del supporto per Containerd 1.x: Kubernetes 1.35 è l'ultima versione che supporta containerd 1.x. È necessario passare a containerd 2.0 o versioni successive prima di eseguire l'aggiornamento alla versione successiva di Kubernetes.
- Nel marzo 2026, il progetto Kubernetes upstream ritirerà Ingress NGINX, un componente dell'infrastruttura fondamentale per molti ambienti Kubernetes.
  - Azione necessaria: i clienti EKS dovrebbero valutare se si affidano a Ingress NGINX e iniziare a pianificare la migrazione verso alternative come l'API Gateway o i controller Ingress di terze parti, poiché non ci saranno ulteriori versioni per correzioni di bug, patch di sicurezza o aggiornamenti dopo il ritiro. Le implementazioni esistenti continueranno a funzionare, ma rimanere con Ingress NGINX dopo il ritiro rende l'ambiente vulnerabile ai rischi di sicurezza, poiché nessuna delle alternative disponibili è una sostituzione diretta e richiederà tempo di pianificazione e progettazione. Per ulteriori informazioni su questo annuncio di Kubernetes, consulta la dichiarazione ufficiale del ritiro di Ingress NGINX da Ingress NGINX da Kubernetes Steering and Security Response Committees. https://kubernetes.io/blog/2026/01/29/ingress-nginx-statement/

- In-Place Pod Resource Updates (stabile): Pod Resource Updates consente agli utenti di regolare le risorse di CPU e memoria senza riavviare  In-Place Pod o contenitori. In precedenza, tali modifiche richiedevano la ricreazione dei Pod, il che poteva interrompere i carichi di lavoro, in particolare per le applicazioni stateful o batch. La nuova funzionalità in-place consente un ridimensionamento verticale più fluido e senza interruzioni, migliora l'efficienza e può anche semplificare lo sviluppo.
  - Per ulteriori informazioni, consulta  Kubernetes 1.35: In-Place Pod Resize Graduates to Stable sul blog di Kubernetes.

- PreferSameNode Distribuzione del traffico (stabile): il `trafficDistribution` campo Servizi è stato aggiornato per fornire un controllo più esplicito sull'instradamento del traffico. È stata introdotta una nuova opzione`PreferSameNode`,, per consentire ai servizi di assegnare una priorità rigorosa agli endpoint sul nodo locale quando disponibile, ricorrendo altrimenti agli endpoint remoti. Questa modifica rende l'API più esplicita sulla preferenza del traffico all'interno del nodo corrente.
- StatefulSet MaxUnavailable (Beta): questa funzionalità consente gli aggiornamenti paralleli dei Pod impostando `maxUnavailable` (ad esempio, 3 o 10%), consentendo alle applicazioni stateful come i cluster di database di aggiornarsi fino al 60% più velocemente rispetto agli aggiornamenti sequenziali uno alla volta, riducendo significativamente le finestre di manutenzione.
- Supporto per Windows Server 2025: EKS 1.35 aggiunge il supporto per Windows Server 2025. https://docs.aws.amazon.com/eks/latest/userguide/eks-optimized-windows-ami.html
- Rimozione della bandiera Kubelet: la **`--pod-infra-container-image`** bandiera è stata rimossa da kubelet. Gli utenti delle AMI personalizzate devono rimuovere questo flag dalla configurazione di Kubelet prima di eseguire l'aggiornamento alla versione 1.35.
- Avviso di deprecazione - Modalità IPVS: la modalità IPVS in kube-proxy è obsoleta e verrà rimossa in Kubernetes 1.36.

Per il `1.35` changelog completo di Kubernetes, vedi https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.35.md

## Kubernetes 1.34

Kubernetes `1.34` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.34`, consulta l'official release announcement.

> **Important**
>
> - Containerd è stato aggiornato alla 2.1 nella versione 1.34 per il lancio.
>   - In caso di problemi dopo l'aggiornamento, consulta le note di rilascio di containerd 2.1.
> - AWS non sta rilasciando un'AMI EKS-optimized Amazon Linux 2 per Kubernetes 1.34.
>   - AWS ti incoraggia a migrare ad Amazon Linux 2023. Informazioni su come the section called "Aggiornamento ad AL2023".
>   - Per ulteriori informazioni, consulta the section called "Deprecazione dell'AMI Amazon Linux 2".
> - AppArmor è deprecato in Kubernetes 1.34.

    - Consigliamo di migrare a soluzioni di sicurezza dei container alternative come seccomp o Pod Security Standards. https://kubernetes.io/docs/tutorials/security/seccomp/ https://kubernetes.io/docs/concepts/security/pod-security-standards/
  - VolumeAttributesClass (VAC) si laurea in GA in Kubernetes 1.34, migrando dall'API beta () all'API stabile (). `storage.k8s.io/v1beta1` `storage.k8s.io/v1`
    - Se si utilizza il driver EBS CSI con container sidecar AWS gestiti (da CSI Components sulla ECR Gallery), la modifica del volume continuerà a funzionare senza problemi sui cluster EKS 1.31-1.33. AWS applicherà delle patch ai sidecar per supportare le API VAC beta fino alla fine del supporto per lo standard EKS 1.33 (29 luglio 2026).
    - Se gestisci autonomamente i contenitori sidecar CSI, potresti dover aggiungere le versioni sidecar precedenti ai cluster precedenti alla 1.34 per mantenere la funzionalità VAC.
    - Per utilizzare le VolumeAttributesClass funzionalità GA (come il rollback delle modifiche), esegui l'aggiornamento a EKS 1.34 o successivo.
  - Il firmatario esterno di JWT Signer for Service Account Tokens viene promosso alla versione beta. Quando si utilizzano firmatari esterni, il flag --service-account-extend-token-expiration non è più completamente rispettato. Il server API impone la scadenza minima tra l'estensione desiderata (1 anno) e il limite del firmatario esterno (24 ore).
    - Consigliamo di utilizzare token di account di servizio associati, che vengono montati e ruotati automaticamente da Kubernetes.

- Dynamic Resource Allocation (DRA) Core API (GA): Dynamic Resource Allocation è diventata stabile, consentendo una gestione efficiente di hardware specializzato come le GPU attraverso interfacce di allocazione standardizzate, semplificando la gestione delle risorse per gli acceleratori hardware e migliorando l'utilizzo di risorse specializzate.
- Projected ServiceAccount Tokens for Kubelet (Beta): questo miglioramento migliora la sicurezza utilizzando credenziali di breve durata per l'estrazione delle immagini dei container anziché segreti di lunga durata, riducendo il rischio di esposizione delle credenziali e rafforzando il livello di sicurezza generale dei cluster.
- Pod-level Resource Requests and Limits (Beta): questa funzionalità semplifica la gestione delle risorse consentendo pool di risorse condivisi per pod multi-container, consentendo un'allocazione e un utilizzo più efficienti delle risorse per applicazioni complesse con più contenitori.
- Mutable CSI Node Allocatable Count (Beta): il `MutableCSINodeAllocatableCount` feature gate è abilitato per impostazione predefinita in EKS 1.34, rendendo mutabile l'attributo CSINode

max attachable volume count e introducendo un meccanismo per aggiornarlo dinamicamente in base alla configurazione utente a livello di driver CSI. Questi aggiornamenti possono essere attivati da intervalli periodici o dal rilevamento di errori, migliorando l'affidabilità della pianificazione stateful dei pod risolvendo le discrepanze tra la capacità di collegamento riportata e quella effettiva sui nodi.

  - Per ulteriori informazioni, consulta Kubernetes v1.34: Mutable CSI Node Allocatable Count nel blog di Kubernetes.

- Avviso di deprecazione - configurazione del driver cgroup: la configurazione manuale del driver cgroup è obsoleta a favore del rilevamento automatico.
  - Impatto sul cliente: se attualmente imposti il `--cgroup-driver` flag manualmente nella configurazione di Kubelet, dovresti prepararti a rimuovere questa configurazione.
  - Azione richiesta: pianifica di aggiornare gli script di bootstrap dei nodi e le configurazioni AMI personalizzate per rimuovere le impostazioni manuali dei driver cgroup prima che la funzionalità venga rimossa in una futura versione di Kubernetes.
  - Per ulteriori informazioni, consulta la documentazione del driver cgroup. https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/configure-cgroup-driver/

Per il changelog completo di Kubernetes`1.34`, consulta https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.34.md

## Dai un'occhiata alle note di rilascio per le versioni Kubernetes sul supporto esteso

Amazon EKS supporta le versioni di Kubernetes più a lungo di quelle upstream, con supporto standard per le versioni secondarie di Kubernetes per 14 mesi dal momento in cui vengono rilasciate in Amazon EKS e supporto esteso per le versioni secondarie di Kubernetes per altri 12 mesi di supporto (26 mesi totali per versione).

Questo argomento fornisce importanti modifiche di cui tenere conto per ogni versione di Kubernetes del supporto esteso. Durante l'aggiornamento, esamina attentamente le modifiche apportate tra la vecchia e la nuova versione del cluster.

> **Note**
>
> Se si esegue il rollback da una versione con supporto standard a una versione con supporto esteso, riprendono i costi del supporto esteso per quel cluster.

## Kubernetes 1.33

Kubernetes `1.33` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.33`, consulta l'official release announcement.

**Important**

- L'API Kubernetes beta dell'allocazione dinamica delle risorse è abilitata.
  - Questa API beta migliora l'esperienza di pianificazione e monitoraggio dei carichi di lavoro che richiedono risorse come le GPU.
  - L'API beta è definita dalla community Kubernetes e potrebbe cambiare nelle versioni future.
  - Esamina attentamente Feature stages nella documentazione di Kubernetes per comprendere le implicazioni dell'utilizzo delle API beta.
- AWS non sta rilasciando un'AMI EKS-optimized Amazon Linux 2 per Kubernetes 1.33.
  - AWS ti incoraggia a migrare ad Amazon Linux 2023. Informazioni su come the section called "Aggiornamento ad AL2023".
  - Per ulteriori informazioni, consulta the section called "Deprecazione dell'AMI Amazon Linux 2".

- In-Place Pod Resource Resize (Beta): Kubernetes 1.33 promuove il ridimensionamento delle risorse sul posto alla versione beta. Ciò consente aggiornamenti dinamici alla CPU e alle risorse di memoria per i Pod esistenti senza riavvii, consentendo il ridimensionamento verticale dei carichi di lavoro statici senza tempi di inattività.
- Contenitori Sidecar ora stabili: i contenitori Sidecar sono diventati stabili. Sono implementati come speciali contenitori init `restartPolicy: Always` che iniziano prima dei contenitori applicativi, funzionano per tutto il ciclo di vita del Pod e supportano le sonde per la segnalazione dello stato operativo.
  - Per ulteriori informazioni, consulta Sidecar Containers nella documentazione Kubernetes.
- Deprecazione dell'API Endpoints: l'API Endpoints è ora ufficialmente obsoleta e restituirà avvisi in caso di accesso: migra i carichi di lavoro e gli script per utilizzare invece l'API, che supporta funzionalità moderne come il networking dual-stack e ne gestisce più per servizio. EndpointSlices EndpointSlices

  - Per ulteriori informazioni, consulta Kubernetes v1.33: Continuare la transizione da Endpoints a sul blog di Kubernetes. EndpointSlice
- Supporto di Elastic Fabric Adapter: il gruppo di sicurezza predefinito per i cluster Amazon EKS ora supporta il traffico Elastic Fabric Adapter (EFA). Il gruppo di sicurezza predefinito ha una nuova regola in uscita che consente il traffico di EFA con la destinazione dello stesso gruppo di sicurezza. Questo consente il traffico EFA all'interno del cluster.
  - Per ulteriori informazioni, consulta Elastic Fabric Adapter for AI/ML and HPC workloads on Amazon EC2 nella Amazon Elastic Compute Cloud User Guide.

Per il changelog completo di Kubernetes, consulta `1.33` https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.33.md

## Kubernetes 1.32

Kubernetes `1.32` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.32`, consulta l'official release announcement.

> **Important**
>
> - La versione `flowcontrol.apiserver.k8s.io/v1beta3` API di FlowSchema ed PriorityLevelConfiguration è stata rimossa nella versione. `1.32` Se utilizzi queste API, devi aggiornare le configurazioni per utilizzare l'ultima versione supportata prima dell'aggiornamento.
> - ServiceAccount `metadata.annotations[kubernetes.io/enforce-mountable-secrets]`è stata deprecata nella versione `1.32` e verrà rimossa in una futura versione minore di Kubernetes. Si consiglia di utilizzare namespace separati per isolare l'accesso ai segreti montati.
> - La versione Kubernetes `1.32` è l'ultima per la quale Amazon EKS rilascerà AMI Amazon Linux 2 (AL2). A partire dalla versione `1.33`, Amazon EKS continuerà a rilasciare AMI basate su Amazon Linux 2023 (AL2023) e Bottlerocket.

- La funzionalità Memory Manager è passata allo stato di disponibilità generale (GA) nella versione Kubernetes `1.32`. Questo miglioramento offre un'allocazione della memoria più efficiente e prevedibile per le applicazioni containerizzate, particolarmente utile per carichi di lavoro con requisiti di memoria specifici.

- PersistentVolumeClaims (PVC) creati ormai includono funzionalità di pulizia automatica. StatefulSets Quando i PVC non saranno più necessari, verranno eliminati automaticamente mantenendo la persistenza dei dati durante StatefulSet gli aggiornamenti e le operazioni di manutenzione dei nodi. Questa funzionalità semplifica la gestione dell'archiviazione e aiuta a prevenire la presenza di PVC orfani nel cluster.
- È stata introdotta la funzionalità Custom Resource Field Selector, che consente agli sviluppatori di aggiungere selettori di campo alle risorse personalizzate. Questa funzionalità offre le stesse funzionalità di filtraggio disponibili per gli oggetti Kubernetes integrati nelle risorse personalizzate, consentendo un filtraggio delle risorse più preciso ed efficiente e promuovendo migliori pratiche di progettazione delle API.

Per il changelog completo di Kubernetes`1.32`, consulta https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.32.md

Modifiche dell'autenticazione anonima

A partire da Amazon EKS `1.32`, l'autenticazione anonima è limitata ai seguenti endpoint di controllo dell'integrità del server API:

- `/healthz`
- `/livez`
- `/readyz`

Le richieste a qualsiasi altro endpoint che utilizza l'utente `system:unauthenticated` riceveranno una risposta HTTP `401 Unauthorized`. Questo miglioramento della sicurezza aiuta a prevenire l'accesso involontario al cluster che potrebbe verificarsi a causa di policy RBAC non configurate correttamente.

> **Note**
>
> Il ruolo RBAC `public-info-viewer` continua a valere per gli endpoint di controllo dell'integrità sopra elencati.

Deprecazione dell'AMI Amazon Linux 2

La versione Kubernetes è l'ultima versione per la quale Amazon `1.32` EKS ha rilasciato le AMI AL2. A partire dalla versione `1.33`, Amazon EKS continuerà a rilasciare AMI basate su AL2023 e Bottlerocket. Per ulteriori informazioni, consulta the section called "Definizione delle AMI AL2 come obsolete".

## Kubernetes 1.31

Kubernetes `1.31` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.31`, consulta l'official release announcement.

> **Important**
>
> - Il flag kubelet `--keep-terminated-pod-volumes` obsoleto dal 2017 è stato rimosso come parte del rilascio della versione `1.31`. Questa modifica influisce sul modo in cui i volumi dei pod terminati vengono gestiti da kubelet. Se si utilizza questo flag nelle configurazioni dei nodi, è necessario aggiornare gli script di bootstrap e avviare i modelli per rimuoverlo prima dell'aggiornamento.

- La funzionalità di controllo `VolumeAttributesClass` beta e la risorsa API sono abilitati nella versione Amazon EKS `1.31`. Questa funzionalità consente agli operatori del cluster di modificare le proprietà mutabili dei volumi persistenti (PV) gestiti da driver CSI compatibili, incluso il driver CSI di Amazon EBS. Per sfruttare questa funzionalità, assicurati che il driver CSI supporti la funzionalità `VolumeAttributesClass` (per il driver CSI di Amazon EBS, esegui l'upgrade alla versione `1.35.0` o successiva per abilitarla automaticamente). Potrai creare oggetti `VolumeAttributesClass` per definire gli attributi di volume desiderati, come il tipo di volume e il throughput, e associarli alle tue richieste di volumi persistenti (PVC). Per ulteriori informazioni, consulta official Kubernetes documentation e la documentazione del driver CSI.
  - Per ulteriori informazioni sul Driver CSI di Amazon EBS, consulta the section called "Amazon EBS".
- Il supporto Kubernetes per AppArmor è diventato stabile ed è ora generalmente disponibile per l'uso pubblico. Questa funzione ti consente di proteggere i tuoi contenitori AppArmor impostando il `appArmorProfile.type` campo in quello del contenitore. `securityContext` Prima della versione Kubernetes`1.30`, AppArmor era controllato da annotazioni. A partire dalla versione `1.30`, è controllato tramite campi. Per sfruttare questa funzionalità, consigliamo di abbandonare

le annotazioni e utilizzare il campo `appArmorProfile.type` per garantire la compatibilità dei carichi di lavoro.

- L' PersistentVolume ultima funzionalità relativa al tempo di transizione di fase è diventata stabile ed è ora generalmente disponibile per l'uso pubblico nella versione Kubernetes. `1.31` Questa funzionalità introduce un nuovo campo, in `.status.lastTransitionTime` PersistentVolumeStatus, che fornisce un timestamp dell' PersistentVolume ultima transizione a una fase diversa. Questo miglioramento consente un migliore monitoraggio e gestione PersistentVolumes, in particolare negli scenari in cui è importante comprendere il ciclo di vita dei volumi.

Per il changelog completo di `1.31` Kubernetes, consulta https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.31.md

## Visualizzazione del periodo di supporto del cluster corrente

La sezione relativa al periodo di supporto per i cluster della console AWS indica se per il cluster è attualmente su un supporto standard o esteso. Se il periodo di supporto del cluster è Supporto esteso, ti sarà addebitato il costo del supporto esteso EKS.

Per ulteriori informazioni sul supporto standard ed esteso, consulta Comprendere il ciclo di vita delle versioni di Kubernetes su EKS.

1. Vai alla pagina Cluster nella sezione EKS della console AWS. Verifica che la console sia impostata sulla stessa regione AWS del cluster che desideri esaminare.
2. Esamina la colonna Periodo di supporto. Se il valore è Supporto standard fino a..., al momento non ti è addebitato alcun costo per il supporto esteso. Rientri nel periodo di supporto standard. Se il valore è Supporto esteso..., al momento a questo cluster è addebitato il costo del supporto esteso.

> **Note**
>
> Il periodo di supporto non può essere recuperato con AWS API o CLI.

## Visualizzazione della corrente policy di aggiornamento del cluster

La policy di aggiornamento del cluster determina cosa succede al cluster quando esce dal periodo di supporto standard. Se la policy di aggiornamento è `EXTENDED`, il cluster non sarà aggiornato

automaticamente ed entrerà il supporto esteso. Se la tua policy di aggiornamento è `STANDARD`, sarà aggiornata automaticamente.

La policy dei controlli delle versioni di Amazon EKS per Kubernetes ti consente di scegliere la fine del comportamento di supporto standard per i tuoi cluster EKS. Con questi controlli è possibile decidere quali cluster debbano essere sottoposti al supporto esteso e quali aggiornati automaticamente alla fine del supporto standard per una versione di Kubernetes.

Una versione secondaria è supportata come standard in Amazon EKS per i primi 14 mesi dopo il rilascio. Una volta superata la data di fine del supporto standard, la versione passa al supporto esteso per i 12 mesi successivi. Il supporto esteso consente di mantenere una versione Kubernetes specifica di più a lungo a un costo aggiuntivo per ora del cluster. Puoi abilitare o disabilitare il supporto esteso per un cluster EKS. Se disabiliti il supporto esteso, AWS aggiornerai automaticamente il cluster alla versione successiva al termine del supporto standard. Se abiliti il supporto esteso, è possibile mantenere la versione corrente a un costo aggiuntivo per un periodo di tempo limitato. Pianifica di aggiornare regolarmente il tuo cluster Kubernetes, anche se utilizzi il supporto esteso.

È possibile impostare la policy della versione sia per i cluster nuovi che per quelli esistenti, utilizzando la proprietà `supportType`. Esistono due opzioni che possono essere utilizzate per impostare la policy di supporto della versione:

- **`STANDARD`** — Il cluster EKS è idoneo per l'aggiornamento automatico al termine del supporto standard. Questa impostazione non prevede costi di supporto esteso, ma il cluster EKS verrà aggiornato automaticamente alla successiva versione di Kubernetes supportata con supporto standard.
- **`EXTENDED`** : il tuo cluster EKS entrerà in supporto esteso una volta che la versione Kubernetes raggiungerà la fine del supporto standard. Con questa impostazione ti saranno addebitati costi di supporto estesi. È possibile aggiornare il tuo cluster a una versione Kubernetes standard supportata per evitare di incorrere in costi di supporto esteso. I cluster che utilizzano il supporto esteso saranno idonei all'aggiornamento automatico al termine del supporto esteso.

Il supporto esteso è abilitato per impostazione predefinita per i nuovi cluster e per i cluster esistenti. È possibile verificare se il supporto esteso è abilitato per un cluster nel Console di gestione AWS, o utilizzando la CLI. AWS

> **Important**
>
> Se desideri che il tuo cluster rimanga nella versione corrente di Kubernetes per sfruttare il periodo di supporto esteso, ti consigliamo vivamente di abilitare la politica di aggiornamento del supporto esteso prima della fine del periodo di supporto standard per la versione Kubernetes del tuo cluster. Puoi abilitare il supporto esteso al termine del periodo di supporto standard. Tuttavia, Amazon EKS non può garantire che la modifica abbia effetto se è già stato avviato un aggiornamento automatico per il cluster.

Una volta che il cluster è entrato nel supporto esteso, non è possibile disabilitarlo. Per modificare l'impostazione della politica di supporto delle versioni, il cluster deve essere in esecuzione su una versione di Kubernetes con supporto standard.

## Visualizza la politica di aggiornamento del cluster (AWS Console)

1. Vai alla  pagina  Clusters nella sezione EKS della AWS Console. Verifica che la console sia impostata sulla stessa AWS regione del cluster che desideri esaminare.
2. Esamina la colonna Policy di aggiornamento. Se il valore è Supporto standard, il cluster non accederà al supporto esteso. Se il valore è Supporto esteso, il cluster entrerà nel supporto esteso.

## Visualizza la politica di aggiornamento del cluster (AWS CLI)

1. Verifica che la AWS CLI sia installata e che tu abbia effettuato l'accesso. Scopri come aggiornare e installare la CLI AWS .
2. Determinare il nome del cluster EKS. Imposta la CLI AWS nella stessa regione del tuo cluster EKS.
3. Esegui il comando seguente:

```
aws eks describe-cluster \
--name <cluster-name> \
--query "cluster.upgradePolicy.supportType"
```

4. Se il valore è `STANDARD`, il tuo cluster non accederà al supporto esteso. Se il valore è `EXTENDED`, il cluster accederà al supporto esteso.

# Aggiungere flessibilità per pianificare gli aggiornamenti delle versioni di Kubernetes abilitando il supporto esteso EKS

Questo argomento descrive come configurare la policy di aggiornamento di un cluster EKS per abilitare il supporto esteso. La policy di aggiornamento di un cluster EKS determina cosa succede quando un cluster raggiunge la fine del periodo di supporto standard. Se una policy di aggiornamento ha il supporto esteso abilitato, alla fine del periodo di supporto standard il cluster entrerà nel periodo di supporto esteso. Alla fine del periodo di supporto standard, il cluster non verrà aggiornato automaticamente.

Vengono addebitati costi più alti per i cluster che si trovano effettivamente nel periodo di supporto esteso. Se la policy di aggiornamento per è solamente configurata per abilitare il supporto esteso, ma il cluster associato si trova nel periodo di supporto standard, vengono addebitati i costi standard.

Se si crea un cluster nella AWS console, verrà impostata la politica di aggiornamento per disabilitare il supporto esteso. Se si crea un cluster con un'altra modalità, la policy di aggiornamento associata sarà configurata per abilitare il supporto esteso. Ad esempio, per i cluster creati con l' AWS API è abilitato il supporto esteso.

Per ulteriori informazioni sulle policy di aggiornamento, consultare la pagina Cluster upgrade policy.

> **Important**
>
> Se desideri che il tuo cluster rimanga sulla versione attuale di Kubernetes per sfruttare il periodo di supporto esteso, ti consigliamo vivamente di abilitare la politica di aggiornamento del supporto esteso prima della fine del periodo di supporto standard per la versione Kubernetes del tuo cluster. Puoi abilitare il supporto esteso al termine del periodo di supporto standard. Tuttavia, Amazon EKS non può garantire che la modifica abbia effetto se è già stato avviato un aggiornamento automatico per il cluster.
> Una volta che il cluster ha ottenuto il supporto esteso, non è possibile disabilitarlo. Puoi disabilitare il supporto esteso solo per i cluster che eseguono una versione di Kubernetes nel supporto standard.
> Se non abiliti il supporto esteso, il cluster verrà aggiornato automaticamente dopo la fine del periodo di supporto standard.

### Abilita il supporto esteso EKS (AWS console)

1. Accedi al tuo cluster EKS nella AWS console. Selezionare la scheda Panoramica nella pagina Informazioni sul cluster.
2. Nella sezione Impostazioni della versione di Kubernetes, selezionare Gestisci.
3. Selezionare Supporto esteso, poi Salva modifiche.

### Abilita il supporto esteso EKS (AWS CLI)

1. Verifica che la AWS CLI sia installata e che tu abbia effettuato l'accesso. Scopri come aggiornare e installare la AWS CLI.
2. Determinare il nome del cluster EKS.
3. Esegui il comando seguente:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=EXTENDED
```

## Prevenire l'aumento del costo dei cluster disabilitando il supporto esteso EKS

Questo argomento descrive come configurare la policy di aggiornamento di un cluster EKS per disabilitare il supporto esteso. La policy di aggiornamento di un cluster EKS determina cosa succede quando un cluster raggiunge la fine del periodo di supporto standard. Se la policy di aggiornamento di un cluster ha il supporto esteso disabilitato, verrà automaticamente aggiornato alla versione successiva di Kubernetes.

Per ulteriori informazioni sulle policy di aggiornamento, consultare la pagina Cluster upgrade policy.

> **Important**
>
> Non è possibile disabilitare il supporto esteso una volta che il cluster vi è entrato. È possibile disabilitare il supporto esteso per i cluster solo durante il supporto standard.
> AWS consiglia di aggiornare il cluster a una versione nel periodo di supporto standard.

### Disattiva il supporto esteso EKS (AWS Console)

1. Accedi al tuo cluster EKS nella AWS console. Selezionare la scheda Panoramica nella pagina Informazioni sul cluster.
2. Nella sezione Impostazioni della versione di Kubernetes, selezionare Gestisci.
3. Selezionare Supporto standard, poi Salva modifiche.

### Disattiva il supporto esteso EKS (AWS CLI)

1. Verifica che la AWS CLI sia installata e che tu abbia effettuato l'accesso. Scopri come aggiornare e installare la AWS CLI.
2. Determinare il nome del cluster EKS.
3. Esegui il comando seguente:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=STANDARD
```

## Visualizzazione delle versioni della piattaforma Amazon EKS per ciascuna versione Kubernetes

Le versioni della piattaforma Amazon EKS rappresentano le funzionalità del piano di controllo del cluster. Ad esempio quali flag del server API di Kubernetes sono abilitati e l'attuale versione della patch di Kubernetes. Ogni versione secondaria Kubernetes dispone di una o più versioni della piattaforma Amazon EKS associate. Le versioni della piattaforma per le diverse versioni secondarie di Kubernetes sono indipendenti. Puoi recuperare la versione corrente della piattaforma del tuo cluster utilizzando la CLI o. AWS Console di gestione AWS Se hai un cluster locale su AWS Outposts, consulta the section called "Versioni della piattaforma EKS" invece questo argomento.

Quando una nuova versione secondaria di Kubernetes è disponibile in Amazon EKS, ad esempio 1.36, la versione iniziale della piattaforma Amazon EKS per quella versione secondaria di Kubernetes parte da. `eks.1` Tuttavia, Amazon EKS rilascia periodicamente nuove versioni della piattaforma per abilitare nuove impostazioni del piano di controllo Kubernetes e fornire soluzioni per problemi relativi alla sicurezza.

Quando nuove versioni della piattaforma Amazon EKS diventano disponibili per una versione secondaria:

- Il numero di versione della piattaforma Amazon EKS viene incrementato (`eks.<n+1>`).
- Amazon EKS aggiorna automaticamente tutti i cluster esistenti alla versione della piattaforma Amazon EKS più recente per la versione secondaria Kubernetes corrispondente. Aggiornamenti automatici di versioni della piattaforma Amazon EKS esistenti vengono implementati in modo incrementale. Il processo di implementazione potrebbe richiedere alcuni minuti. Se le caratteristiche della versione della piattaforma Amazon EKS sono richieste immediatamente, occorre creare un nuovo cluster.

  Se il tuo cluster ha più di due versioni di piattaforma rispetto alla versione corrente della piattaforma, è possibile che Amazon EKS non sia riuscito ad aggiornare automaticamente il cluster. Per i dettagli su cosa potrebbe causare questo problema, consulta the section called "La versione della piattaforma Amazon EKS è più avanti di due versioni rispetto all'attuale versione della piattaforma".
- Amazon EKS potrebbe pubblicare un nuovo nodo AMI con una versione patch corrispondente. Tuttavia, tutte le versioni patch sono compatibili tra il piano di controllo EKS e le AMI dei nodi per una data versione secondaria Kubernetes.

Nuove versioni della piattaforma Amazon EKS non introducono modifiche di conflitto né causano interruzioni del servizio.

I cluster vengono sempre creati con la versione della piattaforma Amazon EKS disponibile più recente (`eks.<n>`) per la versione Kubernetes specificata. Se si aggiorna il cluster a una nuova versione secondaria Kubernetes, il cluster riceverà la versione della piattaforma Amazon EKS corrente per la versione secondaria Kubernetes a cui è stata aggiornata.

Le versioni della piattaforma Amazon EKS correnti e recenti sono descritte nelle tabelle qui di seguito.

> **Note**
>
> AWS recentemente ha disabilitato alcune versioni della piattaforma pubblicate nel giugno 2024. Le versioni della piattaforma presentavano problemi di stabilità. Non è richiesta alcuna azione.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, è possibile iscriversi al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/versioning/platform-versions.adoc.atom
```

## Kubernetes versione 1.36

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma `1.36`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| `1.36.2` | `eks.10` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2026 |
| `1.36.2` | `eks.9` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 7 agosto 2026 |
| `1.36.2` | `eks.8` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 23 luglio 2026 |
| `1.36.2` | `eks.7` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 14 luglio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.36.2 | eks.6 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.36 eks.5 è stata scartata internamente e non è mai stata rilasciata. | 22 giugno 2026 |
| 1.36.1 | eks.4 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 giugno 2026 |
| 1.36.1 | eks.3 | Rilascio iniziale della versione 1.36 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.36". | 2 giugno 2026 |

## Kubernetes versione 1.35

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.35: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.35.6 | eks.21 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2026 |
| 1.35.6 | eks.20 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 7 agosto 2026 |
| 1.35.6 | eks.19 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 23 luglio 2026 |
| 1.35.6 | eks.18 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 14 luglio 2026 |
| 1.35.6 | eks.17 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.35 eks.16 è stata scartata internamente e non è mai stata rilasciata. | 22 giugno 2026 |
| 1.35.5 | eks.15 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 giugno 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.35.5 | eks.14 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 maggio 2026 |
| 1.35.4 | eks.13 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.35 eks.12 è stata scartata internamente e non è mai stata rilasciata. | 19 maggio 2026 |
| 1.35.4 | eks.11 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.35 eks.10 è stata scartata internamente e non è mai stata rilasciata. | 07 maggio 2026 |
| 1.35.2 | eks.9 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 aprile 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.35.2 | eks.8 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La 1.35 e la 1.35 sono state eks.7 scartate eks.6 internamente e non sono mai state rilasciate. | 18 marzo 2026 |
| 1.35.0 | eks.5 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 marzo 2026 |
| 1.35.0 | eks.4 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2026 |
| 1.35.0 | eks.3 | Rilascio iniziale della versione 1.35 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.35". | 27 gennaio 2026 |

## Kubernetes versione 1.34

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.34: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection,

`PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| `1.34.9` | `eks.31` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2026 |
| `1.34.9` | `eks.30` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 7 agosto 2026 |
| `1.34.9` | `eks.29` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 23 luglio 2026 |
| `1.34.9` | `eks.28` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 14 luglio 2026 |
| `1.34.9` | `eks.27` | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.34 `eks.26` è stata scartata internamente e non è mai stata rilasciata. | 22 giugno 2026 |
| `1.34.8` | `eks.25` | Nuova versione della piattaforma con | 3 giugno 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | correzioni di sicurezza e miglioramenti. | |
| 1.34.8 | eks.24 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 maggio 2026 |
| 1.34.7 | eks.23 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.34 eks.22 è stata scartata internamente e non è mai stata rilasciata. | 19 maggio 2026 |
| 1.34.7 | eks.21 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.34 eks.20 è stata scartata internamente e non è mai stata rilasciata. | 07 maggio 2026 |
| 1.34.4 | eks.19 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 aprile 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.34.4 | eks.18 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. 1.34 e 1.34 sono stati eks.17 scartati internamente eks.16 e non sono mai stati rilasciati. | 18 marzo 2026 |
| 1.34.3 | eks.15 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 marzo 2026 |
| 1.34.3 | eks.14 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2026 |
| 1.34.3 | eks.13 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 gennaio 2026 |
| 1.34.3 | eks.12 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.34 eks.11 è stata scartata internamente e non è mai stata rilasciata. | 16 gennaio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.34.2 | eks.10 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 19 dicembre 2025 |
| 1.34.1 | eks.9 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 novembre 2025 |
| 1.34.1 | eks.8 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 novembre 2025 |
| 1.34.1 | eks.7 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 6 novembre 2025 |
| 1.34.1 | eks.6 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 ottobre 2025 |
| 1.34.1 | eks.5 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.34.1 | eks.4 | Rilascio iniziale della versione 1.34 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.34". | 2 ottobre 2025 |

## Kubernetes versione 1.33

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.33: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.33.13 | eks.45 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2026 |
| 1.33.13 | eks.44 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 7 agosto 2026 |
| 1.33.13 | eks.43 | Nuova versione della piattaforma con | 23 luglio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | correzioni di sicurezza e miglioramenti. | |
| 1.33.13 | eks.42 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 14 luglio 2026 |
| 1.33.13 | eks.41 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La 1.33 eks.40 è stata scartata internamente e non è mai stata rilasciata. | 22 giugno 2026 |
| 1.33.12 | eks.39 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 giugno 2026 |
| 1.33.12 | eks.38 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 maggio 2026 |
| 1.33.11 | eks.37 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La 1.33 eks.36 è stata scartata internamente e non è mai stata rilasciata. | 19 maggio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.33.11 | eks.35 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La 1.33 eks.34 è stata scartata internamente e non è mai stata rilasciata. | 07 maggio 2026 |
| 1.33.8 | eks.33 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 aprile 2026 |
| 1.33.8 | eks.32 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.30 1.33 e eks.31 1.33 sono state scartate internamente e non sono mai state rilasciate. | 18 marzo 2026 |
| 1.33.7 | eks.29 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 marzo 2026 |
| 1.33.7 | eks.28 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.33.7 | eks.27 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 gennaio 2026 |
| 1.33.7 | eks.26 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La 1.33 eks.25 è stata scartata internamente e non è mai stata rilasciata. | 16 gennaio 2026 |
| 1.33.6 | eks.24 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 19 dicembre 2025 |
| 1.33.5 | eks.23 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 novembre 2025 |
| 1.33.5 | eks.22 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 novembre 2025 |
| 1.33.5 | eks.21 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 6 novembre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.33.5 | eks.20 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 ottobre 2025 |
| 1.33.5 | eks.19 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 ottobre 2025 |
| 1.33.5 | eks.18 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |
| 1.33.5 | eks.17 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |
| 1.33.5 | eks.16 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 ottobre 2025 |
| 1.33.4 | eks.15 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |
| 1.33.4 | eks.14 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.33.4 | eks.13 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 16 settembre 2025 |
| 1.33.4 | eks.12 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 settembre 2025 |
| 1.33.3 | eks.11 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 25 agosto 2025 |
| 1.33.3 | eks.10 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2025 |
| 1.33.2 | eks.9 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.7 1.33 e eks.8 1.33 sono state scartate internamente e non sono mai state rilasciate. | 30 luglio 2025 |
| 1.33.1 | eks.6 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 giugno 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.33.1 | eks.5 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 11 giugno 2025 |
| 1.33.1 | eks.4 | Rilascio iniziale della versione 1.33 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.33". | 28 maggio 2025 |

## Kubernetes versione 1.32

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.32: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.13 | eks.52 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2026 |
| 1.32.13 | eks.51 | Nuova versione della piattaforma con | 7 agosto 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | correzioni di sicurezza e miglioramenti. | |
| 1.32.13 | eks.50 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 23 luglio 2026 |
| 1.32.13 | eks.49 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 14 luglio 2026 |
| 1.32.13 | eks.48 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.32 eks.47 è stata scartata internamente e non è mai stata rilasciata. | 22 giugno 2026 |
| 1.32.13 | eks.46 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 giugno 2026 |
| 1.32.13 | eks.45 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 maggio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.13 | eks.44 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.32 eks.43 è stata scartata internamente e non è mai stata rilasciata. | 19 maggio 2026 |
| 1.32.13 | eks.42 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.32 eks.41 è stata scartata internamente e non è mai stata rilasciata. | 07 maggio 2026 |
| 1.32.12 | eks.40 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 aprile 2026 |
| 1.32.12 | eks.39 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.37 1.32 e eks.38 1.32 sono state scartate internamente e non sono mai state rilasciate. | 18 marzo 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.11 | eks.36 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 marzo 2026 |
| 1.32.11 | eks.35 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2026 |
| 1.32.11 | eks.34 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 gennaio 2026 |
| 1.32.11 | eks.33 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.32 eks.32 è stata scartata internamente e non è mai stata rilasciata. | 16 gennaio 2026 |
| 1.32.10 | eks.31 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 19 dicembre 2025 |
| 1.32.9 | eks.30 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 novembre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.9 | eks.29 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 novembre 2025 |
| 1.32.9 | eks.28 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 6 novembre 2025 |
| 1.32.9 | eks.27 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 ottobre 2025 |
| 1.32.9 | eks.26 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 ottobre 2025 |
| 1.32.9 | eks.25 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |
| 1.32.9 | eks.24 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |
| 1.32.9 | eks.23 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 ottobre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.8 | eks.22 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |
| 1.32.8 | eks.21 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |
| 1.32.8 | eks.20 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 16 settembre 2025 |
| 1.32.8 | eks.19 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 settembre 2025 |
| 1.32.7 | eks.18 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 25 agosto 2025 |
| 1.32.7 | eks.17 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.6 | eks.16 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.14 1.32 e eks.15 1.32 sono state scartate internamente e non sono mai state rilasciate. | 30 luglio 2025 |
| 1.32.5 | eks.13 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 giugno 2025 |
| 1.32.5 | eks.12 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 11 giugno 2025 |
| 1.32.5 | eks.11 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 maggio 2025 |
| 1.32.3 | eks.10 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 16 maggio 2025 |
| 1.32.3 | eks.9 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 aprile 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.3 | eks.8 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 aprile 2025 |
| 1.32.3 | eks.7 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 aprile 2025 |
| 1.32.3 | eks.6 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 aprile 2025 |
| 1.32.2 | eks.5 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 marzo 2025 |
| 1.32.2 | eks.4 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 4 marzo 2025 |
| 1.32.1 | eks.3 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.32.0 | eks.2 | Rilascio iniziale della versione 1.32 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.32". | Gennaio 2025 |

## Kubernetes versione 1.31

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.31: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota, ObjectCount.

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.14 | eks.68 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2026 |
| 1.31.14 | eks.67 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 7 agosto 2026 |
| 1.31.14 | eks.66 | Nuova versione della piattaforma con | 23 luglio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| | | correzioni di sicurezza e miglioramenti. | |
| 1.31.14 | eks.65 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 14 luglio 2026 |
| 1.31.14 | eks.64 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.31 eks.63 è stata scartata internamente e non è mai stata rilasciata. | 22 giugno 2026 |
| 1.31.14 | eks.62 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 giugno 2026 |
| 1.31.14 | eks.61 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 maggio 2026 |
| 1.31.14 | eks.60 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.31 eks.59 è stata scartata internamente e non è mai stata rilasciata. | 19 maggio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.14 | eks.58 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.31 eks.57 è stata scartata internamente e non è mai stata rilasciata. | 07 maggio 2026 |
| 1.31.14 | eks.56 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 aprile 2026 |
| 1.31.14 | eks.55 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.53 1.31 e eks.54 1.31 sono state scartate internamente e non sono mai state rilasciate. | 18 marzo 2026 |
| 1.31.14 | eks.52 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 marzo 2026 |
| 1.31.14 | eks.51 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.14 | eks.50 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 gennaio 2026 |
| 1.31.14 | eks.49 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.31 eks.48 è stata scartata internamente e non è mai stata rilasciata. | 16 gennaio 2026 |
| 1.31.14 | eks.47 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 19 dicembre 2025 |
| 1.31.13 | eks.46 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 novembre 2025 |
| 1.31.13 | eks.45 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 novembre 2025 |
| 1.31.13 | eks.44 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 6 novembre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.13 | eks.43 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 ottobre 2025 |
| 1.31.13 | eks.42 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 ottobre 2025 |
| 1.31.13 | eks.41 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |
| 1.31.13 | eks.40 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |
| 1.31.13 | eks.39 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 ottobre 2025 |
| 1.31.12 | eks.38 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |
| 1.31.12 | eks.37 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.12 | eks.36 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 16 settembre 2025 |
| 1.31.12 | eks.35 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 settembre 2025 |
| 1.31.11 | eks.34 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 25 agosto 2025 |
| 1.31.11 | eks.33 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2025 |
| 1.31.10 | eks.32 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.30 1.31 e eks.31 1.31 sono state scartate internamente e non sono mai state rilasciate. | 30 luglio 2025 |
| 1.31.9 | eks.29 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 giugno 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.9 | eks.28 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 11 giugno 2025 |
| 1.31.9 | eks.27 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 maggio 2025 |
| 1.31.7 | eks.26 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 16 maggio 2025 |
| 1.31.7 | eks.25 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 aprile 2025 |
| 1.31.7 | eks.24 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 aprile 2025 |
| 1.31.7 | eks.23 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 aprile 2025 |
| 1.31.7 | eks.22 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 aprile 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.6 | eks.21 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 marzo 2025 |
| 1.31.6 | eks.20 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 4 marzo 2025 |
| 1.31.5 | eks.19 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2025 |
| 1.31.5 | eks.18 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2025 |
| 1.31.4 | eks.17 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 gennaio 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.31.2 | eks.12 | Nuova versione della piattaforma con supporto per Amazon EKS Hybrid Nodes e miglioramenti per controllare l'osservabilità del piano di controllo. Scopri the section called "Nodi ibridi" e consulta Amazon EKS migliora l'osservabilità delle prestazioni, rispettivamente. | 15 novembre 2024 |
| 1.31.1 | eks.6 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 21 ottobre 2024 |
| 1.31.0 | eks.4 | Rilascio iniziale della versione 1.31 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.31". | 26 settembre 2024 |

## Kubernetes versione 1.30

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.30: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority,

`DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| `1.30.14` | `eks.76` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2026 |
| `1.30.14` | `eks.75` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 7 agosto 2026 |
| `1.30.14` | `eks.74` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 23 luglio 2026 |
| `1.30.14` | `eks.73` | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 14 luglio 2026 |
| `1.30.14` | `eks.72` | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.30 `eks.71` è stata scartata internamente e non è mai stata rilasciata. | 22 giugno 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.14 | eks.70 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 giugno 2026 |
| 1.30.14 | eks.69 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 maggio 2026 |
| 1.30.14 | eks.68 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.30 eks.67 è stata scartata internamente e non è mai stata rilasciata. | 19 maggio 2026 |
| 1.30.14 | eks.66 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.30 eks.65 è stata scartata internamente e non è mai stata rilasciata. | 07 maggio 2026 |
| 1.30.14 | eks.64 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 aprile 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.14 | eks.63 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.61 1.30 e eks.62 1.30 sono state scartate internamente e non sono mai state rilasciate. | 18 marzo 2026 |
| 1.30.14 | eks.60 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 marzo 2026 |
| 1.30.14 | eks.59 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2026 |
| 1.30.14 | eks.58 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 gennaio 2026 |
| 1.30.14 | eks.57 | Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. La versione 1.30 eks.56 è stata scartata internamente e non è mai stata rilasciata. | 16 gennaio 2026 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.14 | eks.55 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 19 dicembre 2025 |
| 1.30.14 | eks.54 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 novembre 2025 |
| 1.30.14 | eks.53 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 novembre 2025 |
| 1.30.14 | eks.52 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 6 novembre 2025 |
| 1.30.14 | eks.51 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 ottobre 2025 |
| 1.30.14 | eks.50 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 ottobre 2025 |
| 1.30.14 | eks.49 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.14 | eks.48 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 15 ottobre 2025 |
| 1.30.14 | eks.47 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 ottobre 2025 |
| 1.30.14 | eks.46 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |
| 1.30.14 | eks.45 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 settembre 2025 |
| 1.30.14 | eks.44 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 16 settembre 2025 |
| 1.30.14 | eks.43 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 5 settembre 2025 |
| 1.30.14 | eks.42 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 25 agosto 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.14 | eks.41 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 agosto 2025 |
| 1.30.14 | eks.40 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.38 1.30 e eks.39 1.30 sono state scartate internamente e non sono mai state rilasciate. | 30 luglio 2025 |
| 1.30.13 | eks.37 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 26 giugno 2025 |
| 1.30.13 | eks.36 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 11 giugno 2025 |
| 1.30.13 | eks.35 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 30 maggio 2025 |
| 1.30.11 | eks.34 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 16 maggio 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.11 | eks.33 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 29 aprile 2025 |
| 1.30.11 | eks.32 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 aprile 2025 |
| 1.30.11 | eks.31 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 18 aprile 2025 |
| 1.30.11 | eks.30 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 aprile 2025 |
| 1.30.10 | eks.29 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 marzo 2025 |
| 1.30.10 | eks.28 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 4 marzo 2025 |
| 1.30.9 | eks.27 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 24 febbraio 2025 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.8 | eks.25 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 17 gennaio 2025 |
| 1.30.8 | eks.24 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 gennaio 2025 |
| 1.30.7 | eks.23 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 13 dicembre 2024 |
| 1.30.6 | eks.22 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 13 dicembre 2024 |
| 1.30.6 | eks.21 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 13 dicembre 2024 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.6 | eks.20 | Nuova versione della piattaforma con supporto per Amazon EKS Hybrid Nodes e miglioramenti per controllare l'osservabilità del piano di controllo. Scopri the section called "Nodi ibridi" e consulta Amazon EKS migliora l'osservabilità delle prestazioni, rispettivamente. | 15 novembre 2024 |
| 1.30.5 | eks.12 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 21 ottobre 2024 |
| 1.30.4 | eks.8 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 3 settembre 2024 |
| 1.30.3 | eks.7 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 28 agosto 2024 |
| 1.30.3 | eks.6 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 9 agosto 2024 |

| Versione di Kubernetes | Versioni della piattaforma EKS | Note di rilascio | Data di rilascio |
|---|---|---|---|
| 1.30.2 | eks.5 | Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. | 2 luglio 2024 |
| 1.30.0 | eks.2 | Rilascio iniziale della versione 1.30 di Kubernetes per EKS. | 23 maggio 2024 |

# Ottieni versione corrente della piattaforma

## Ottieni la versione attuale della piattaforma (AWS Console)

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione scegliere Cluster.
3. Nell'elenco dei cluster, scegli il Nome del cluster di cui verificare la versione della piattaforma.
4. Seleziona la scheda Panoramica.
5. La versione della piattaforma è disponibile nella sezione Dettagli.

## Ottieni la versione attuale della piattaforma (AWS CLI)

1. Determina il nome del cluster di cui vuoi controllare la versione della piattaforma.
2. Esegui il comando seguente:

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

Di seguito viene riportato un output di esempio.

```
"eks.10"
```

# Modificare la versione della piattaforma

Non è possibile modificare la versione della piattaforma di un cluster EKS. Quando una nuova versione della piattaforma Amazon EKS diventa disponibile per una versione Kubernetes, EKS aggiorna automaticamente tutti i cluster esistenti all'ultima versione della piattaforma di Amazon EKS per la versione Kubernetes corrispondente. Aggiornamenti automatici di versioni della piattaforma Amazon EKS esistenti vengono implementati in modo incrementale. Non è possibile utilizzare la AWS Console o la CLI per modificare la versione della piattaforma.

Se aggiorni la tua versione Kubernetes, il cluster passerà alla versione della piattaforma più recente per la versione Kubernetes.

# Risoluzione dei problemi con i cluster e i nodi Amazon EKS

Questo capitolo presenta alcuni errori comuni che si potrebbero verificare durante l'uso di Amazon EKS e il modo in cui gestirli. Se hai bisogno di risolvere problemi in aree specifiche di Amazon EKS, consulta gli argomenti separati the section called "Risoluzione dei problemi" e Risoluzione dei problemi per ADOT using EKS. the section called "Risoluzione dei problemi del connettore EKS" Add-Ons

Per altre informazioni sulla risoluzione dei problemi, consulta i contenuti del Knowledge Center su Amazon Elastic Kubernetes Service su RE:POST. AWS

## Capacità insufficiente

Se ricevi il seguente errore mentre provi a creare un cluster Amazon EKS, significa che una delle zone di disponibilità indicate non dispone della capacità sufficiente per supportare un cluster.

```
Cannot create cluster 'example-cluster' because region-1d, the targeted
Availability Zone, does not currently have sufficient capacity to support
the cluster. Retry and choose from these Availability Zones: region-1a,
region-1b, region-1c
```

Riprova a crearlo con le sottoreti del VPC del cluster ospitate nelle zone di disponibilità restituite da questo messaggio di errore.

Ci sono zone di disponibilità dove non è possibile farvi risiedere un cluster. Confronta le zone di disponibilità in cui si trovano le tue sottoreti con l'elenco delle zone di disponibilità in Requisiti e considerazioni sulle sottoreti.

## Impossibile aggiungere i nodi al cluster

Ci sono diversi motivi comuni che impediscono l'aggiunta dei nodi di lavoro al cluster:

- Se i nodi sono gestiti, Amazon EKS aggiunge voci ad `aws-auth ConfigMap` quando crei il gruppo di nodi. Se la voce è stata rimossa o modificata, è necessario aggiungerla nuovamente. Per ulteriori informazioni, digita `eksctl create iamidentitymapping --help` nel terminale. Puoi visualizzare le `aws-auth ConfigMap` voci correnti *`my-cluster`* sostituendo il seguente comando con il nome del tuo cluster e quindi eseguendo il comando modificato:. `eksctl get iamidentitymapping --cluster` *`my-cluster`* L'ARN del ruolo specificato non può

includere un percorso diverso da `/`. Ad esempio, se il nome del ruolo è `development/apps/my-role`, è necessario modificarlo in `my-role` quando si specifica l'ARN per tale ruolo. Assicurati di specificare l'ARN del ruolo IAM del nodo (non l'ARN del profilo dell'istanza).

Se i nodi sono autogestiti e non hai creato voci di accesso per l'ARN del ruolo IAM del nodo, esegui gli stessi comandi elencati per i nodi gestiti. Se hai creato una voce di accesso per l'ARN del tuo ruolo IAM del nodo, è possibile che non sia configurata correttamente nella suddetta voce di accesso. Assicurati che l'ARN del ruolo IAM del nodo (e non l'ARN del profilo dell'istanza) sia specificato come ARN principale nella tua voce `aws-auth ConfigMap` o voce di accesso. Per ulteriori informazioni sulle voci di accesso, consulta la sezione the section called "Voci di accesso".

- Il AWS CloudFormation modello presente `ClusterName` nel tuo nodo non corrisponde esattamente al nome del cluster a cui desideri che i tuoi nodi si uniscano. Se inserisci un valore non corretto in questo campo, si verificherà una configurazione errata del file `/var/lib/kubelet/kubeconfig` del nodo e i nodi non saranno aggiunti al cluster.
- Il nodo non è taggato come appartenente al cluster. Ai nodi deve essere applicato il tag seguente, dove *`my-cluster`* viene sostituito con il nome del cluster.

| Chiave | Valore |
| --- | --- |
| `kubernetes.io/cluster/` *`my-cluster`* | owned |

- I nodi potrebbero non essere in grado di accedere al cluster utilizzando un indirizzo IP pubblico. Assicurarsi che ai nodi implementati nelle sottoreti pubbliche venga assegnato un indirizzo IP pubblico. In caso contrario, è possibile associare un indirizzo IP elastico a un nodo dopo l'avvio. Per ulteriori informazioni, consulta Associazione di un indirizzo IP elastico a un'istanza o a un'interfaccia di rete in esecuzione. Se la sottorete pubblica non è impostata per assegnare automaticamente gli indirizzi IP pubblici alle istanze implementate, è consigliabile abilitare l'impostazione. Per ulteriori informazioni, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete. Se il nodo viene implementato in una sottorete privata, la sottorete deve disporre di una route a un gateway NAT a cui è assegnato un indirizzo IP pubblico.
- L'endpoint AWS STS per la AWS regione in cui stai distribuendo i nodi non è abilitato per il tuo account. Per abilitare la regione, vedi Attivazione e disattivazione AWS di STS in una regione. AWS
- Il nodo non dispone di una voce DNS privata, con conseguente log `kubelet` che riporta un errore `node "" not found`. Assicurati che il VPC in cui viene creato il nodo disponga di valori

impostati per `domain-name` e `domain-name-servers` come `Options` in un `DHCP options set`. I valori predefiniti sono `domain-name:<region>.compute.internal` e `domain-name-servers:AmazonProvidedDNS`. Per ulteriori informazioni, consulta Set opzioni DHCP nella Guida per l'utente di Amazon VPC.

- Se i nodi del gruppo di nodi gestiti non si connettono al cluster entro 15 minuti, verrà emesso un problema di integrità pari a NodeCreationFailure "e lo stato della console verrà impostato su. `Create failed` Per le AMI Windows con tempi di avvio lenti, questo problema può essere risolto utilizzando l'avvio rapido.

Per identificare e risolvere i problemi più comuni che impediscono ai nodi worker di unirsi a un cluster, puoi utilizzare il runbook `AWSSupport-TroubleshootEKSWorkerNode`. Per ulteriori informazioni, vedere `AWSSupport-TroubleshootEKSWorkerNode` nel runbook di riferimento AWS di Systems Manager Automation.

## Accesso non autorizzato o negato (kubectl)

Se ricevi uno dei seguenti errori durante l'esecuzione di comandi `kubectl`, `kubectl` non è stato configurato correttamente per Amazon EKS o le credenziali del principale IAM (ruolo o utente) in uso non sono mappate a un nome utente Kubernetes con autorizzazioni sufficienti per gli oggetti Kubernetes nel tuo cluster Amazon EKS.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn't have a resource type "svc"`

Questo potrebbe essere dovuto a uno dei seguenti fattori:

- Il cluster è stato creato con le credenziali per un principale IAM e `kubectl` è configurato per utilizzare le credenziali per un altro principale IAM. Per risolvere il problema, aggiorna il file `kube config` affinché vengano utilizzate le credenziali con cui è stato creato il cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".
- Se il cluster soddisfa i requisiti minimi di piattaforma indicati nella sezione dei prerequisiti di Concedere ad utenti IAM un accesso a Kubernetes con voci di accesso EKS, non esiste una voce di accesso con il tuo principale IAM. Se esiste, non ha i nomi dei gruppi Kubernetes necessari definiti per essa, o non la policy di accesso corretta non vi è associata. Per ulteriori informazioni, consulta the section called "Voci di accesso".

- Se il tuo cluster non soddisfa i requisiti minimi di piattaforma indicati in Concedere ad utenti IAM un accesso a Kubernetes con voci di accesso EKS, non esiste una voce con il tuo principale IAM in `ConfigMap` `aws-auth`. Se esiste, non è mappata su nomi dei gruppi Kubernetes associati a `Role` o a `ClusterRole` di Kubernetes con le autorizzazioni necessarie. Per ulteriori informazioni sugli oggetti dell'autorizzazione basata sui ruoli (RBAC) di Kubernetes, consulta Utilizzare l'autorizzazione RBAC nella documentazione Kubernetes. È possibile visualizzare le `aws-auth` `ConfigMap` voci correnti *`my-cluster`* sostituendo il comando seguente con il nome del cluster e quindi eseguendo il comando modificato:. `eksctl get iamidentitymapping --cluster` *`my-cluster`* Se una voce con l'ARN del tuo principale IAM non è presente`ConfigMap`, entra `eksctl create iamidentitymapping --help` nel tuo terminale per scoprire come crearne una.

Se installi e configuri la AWS CLI, puoi configurare le credenziali IAM che utilizzi. Per ulteriori informazioni, consulta Configurazione della AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando. È anche possibile configurare `kubectl` affinché utilizzi un ruolo IAM, se assumi un ruolo IAM per accedere agli oggetti Kubernetes sul cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".

## il nome host non corrisponde

La versione Python del sistema deve essere `2.7.9` o successiva. Altrimenti, ricevi `hostname doesn't match` errori con le chiamate AWS CLI ad Amazon EKS. Per ulteriori informazioni, consulta Cosa sono gli errori "hostname non corrispondente"? nelle Domande frequenti di Python Requests.

## getsockopt: nessun percorso verso l'host

Docker viene eseguito nell'intervallo CIDR `172.17.0.0/16` nei cluster Amazon EKS. Consigliamo che le sottoreti VPC del cluster non si sovrappongano a questo intervallo. In caso contrario, riceverai il seguente messaggio di errore:

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

# Le istanze non sono riuscite a entrare nel cluster Kubernetes

Se ricevi l'errore `Instances failed to join the Kubernetes cluster` in Console di gestione AWS, assicurati che l'accesso privato all'endpoint del cluster sia abilitato o di aver configurato correttamente i blocchi CIDR per l'accesso agli endpoint pubblici. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

## Codici di errore dei gruppi di nodi gestiti

Se il gruppo di nodi gestiti rileva un problema di integrità hardware, Amazon EKS restituisce un codice di errore che consente di diagnosticare il problema. Questi controlli dell'integrità non rilevano problemi software perché si basano su controlli dell'integrità di Amazon EC2. Nel seguente elenco vengono descritti i codici di errore.

AccessDenied

Amazon EKS o uno o più nodi gestiti non sono in grado di eseguire l'autenticazione o l'autorizzazione con il server API del cluster Kubernetes. Per ulteriori informazioni sulla risoluzione di un problema comune, consulta the section called "Risoluzione di una causa comune di errori `AccessDenied` per i gruppi di nodi gestiti". Le AMI Windows private possono causare anche questo codice di errore insieme al messaggio di errore `Not authorized for images`. Per ulteriori informazioni, consulta the section called "`Non autorizzato per le immagini`".

AmiIdNotFound

Non è stato possibile trovare l'ID AMI associato al modello di avvio. Assicurati che l'AMI esista e sia condivisa con l'account.

AutoScalingGroupNotFound

Non è stato possibile trovare il gruppo Auto Scaling associato al gruppo di nodi gestiti. Potresti ricreare un gruppo con dimensionamento automatico con le stesse impostazioni per procedere con il ripristino.

ClusterUnreachable

Amazon EKS o uno o più nodi gestiti non sono in grado di comunicare con il server API del cluster Kubernetes. Ciò può verificarsi in caso di interruzioni di rete o se i server API stanno eseguendo il timeout delle richieste di elaborazione.

Ec2SecurityGroupNotFound

Non è stato possibile trovare il gruppo di sicurezza per il cluster. È necessario ricreare il cluster.

Ec2SecurityGroupDeletionFailure

Non è stato possibile eliminare il gruppo di sicurezza dell'accesso remoto per il gruppo di nodi gestiti. Rimuovi eventuali dipendenze dal gruppo di sicurezza.

Ec2LaunchTemplateNotFound

Non è stato possibile trovare il modello di avvio Amazon EC2 per il gruppo di nodi gestiti. È necessario ricreare il gruppo di nodi per procedere con il ripristino.

Ec2LaunchTemplateVersionMismatch

La versione del modello di avvio Amazon EC2 per il gruppo di nodi gestiti non corrisponde alla versione creata da Amazon EKS. Puoi regredire alla versione creata da Amazon EKS per il ripristino.

IamInstanceProfileNotFound

Non è stato possibile trovare il profilo dell'istanza IAM per il gruppo di nodi gestiti. Potresti ricreare un profilo dell'istanza con le stesse impostazioni per il ripristino.

IamNodeRoleNotFound

Non è stato possibile trovare il ruolo IAM per il gruppo di nodi gestiti. Potresti ricreare un ruolo IAM con le stesse impostazioni per il ripristino.

AsgInstanceLaunchFailures

Il gruppo con dimensionamento automatico sta riscontrando errori nel tentativo di avviare le istanze.

NodeCreationFailure

Le istanze avviate non sono in grado di registrarsi con il cluster Amazon EKS. Le cause comuni di questo errore sono le autorizzazioni insufficienti del ruolo IAM del nodo o la mancanza di accesso a Internet in uscita per i nodi. I nodi devono soddisfare uno dei requisiti seguenti:

- Devono essere in grado di accedere a Internet utilizzando un indirizzo IP pubblico. Il gruppo di sicurezza associato alla sottorete in cui si trova il nodo deve consentire la comunicazione. Per ulteriori informazioni, consulta the section called "Considerazioni e requisiti relativi alle sottoreti" e the section called "Requisiti relativi al gruppo di sicurezza".

- I nodi e il VPC devono soddisfare i requisiti in Implementare cluster privato con accesso Internet limitato.

InstanceLimitExceeded

Il tuo AWS account non è in grado di avviare altre istanze del tipo di istanza specificato. Potresti richiedere un aumento del limite di istanze Amazon EC2 per il ripristino.

InsufficientFreeAddresses

Una o più sottoreti associate al gruppo di nodi gestiti non dispone di indirizzi IP disponibili sufficienti per i nuovi nodi.

InternalFailure

Questi errori sono in genere causati da un problema lato server Amazon EKS.

## Risoluzione di una causa comune di errori **`AccessDenied`** per i gruppi di nodi gestiti

La causa più comune degli errori `AccessDenied` durante l'esecuzione di operazioni su gruppi di nodi gestiti è la mancanza di `eks:node-manager` `ClusterRole` o `ClusterRoleBinding`. Amazon EKS imposta queste risorse nel cluster come parte dell'onboarding con i gruppi di nodi gestiti e queste sono necessarie per la gestione dei gruppi di nodi.

Il `ClusterRole` può cambiare nel tempo, ma dovrebbe essere simile all'esempio seguente:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
```

```
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

Il `ClusterRoleBinding` può cambiare nel tempo, ma dovrebbe essere simile all'esempio seguente:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Verifica che il `eks:node-manager` `ClusterRole` esista.

```
kubectl describe clusterrole eks:node-manager
```

Se presente, confrontare l'output con il precedente esempio `ClusterRole`.

Verifica che il `eks:node-manager` `ClusterRoleBinding` esista.

```
kubectl describe clusterrolebinding eks:node-manager
```

Se presente, confrontare l'output con il precedente esempio `ClusterRoleBinding`.

Se è identificato un oggetto mancante o corrotto `ClusterRole` o `ClusterRoleBinding` come causa di un errore `AccessDenied` durante la richiesta di operazioni di gruppo di nodi gestiti, è

possibile ripristinarle. Salva nel computer i seguenti contenuti in un file denominato *eks-node-manager-role.yaml*.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
```

```
  kind: User
  name: eks:node-manager
```

Applica il file.

```
kubectl apply -f eks-node-manager-role.yaml
```

Riprova l'operazione per il gruppo di nodi per verificare se il problema è stato risolto.

## Non autorizzato per le immagini

Una potenziale causa di un messaggio di errore `Not authorized for images` è l'utilizzo di un'AMI Windows di Amazon EKS privata per avviare gruppi di nodi gestiti da Windows. Dopo aver rilasciato nuove AMI Windows, AWS rende private le AMI più vecchie di 4 mesi, rendendole quindi non più accessibili. Se il tuo gruppo di nodi gestiti utilizza un'AMI Windows privata, considera la possibilità di aggiornare il gruppo di nodi gestito da Windows. Sebbene non possiamo garantire di poter fornire l'accesso alle AMI che sono state rese private, puoi richiedere l'accesso presentando un ticket all'assistenza. AWS Per maggiori informazioni, consulta Patches nella Guida per l'utente di Amazon EC2.

## Il nodo è in stato `NotReady`

Se il nodo entra in uno stato `NotReady`, ciò probabilmente indica che il nodo non è integro e non è disponibile per pianificare nuovi pod. Ciò può verificarsi per vari motivi, ad esempio se il nodo non dispone di risorse sufficienti per CPU, memoria o spazio disponibile su disco.

Per le AMI Windows ottimizzate per Amazon EKS, non è prevista alcuna prenotazione per le risorse di elaborazione specificate per impostazione predefinita nella configurazione `kubelet`. Per aiutare a prevenire problemi relativi alle risorse, puoi riservare le risorse di calcolo per i processi di sistema fornendo i valori di configurazione per `kubelet` kube-reserved system-reserved. and/or https://kubernetes.io/docs/tasks/administer-cluster/reserve-compute-resources/#system-reserved È possibile eseguire questa operazione utilizzando il parametro della riga di comando `-KubeletExtraArgs` nello script di bootstrap. Per ulteriori informazioni, consulta Reserve Compute Resources for System Daemons nella documentazione di Kubernetes e the section called "Parametri di configurazione dello script di bootstrap" in questa guida per l'utente.

# Raccoglitore di log EKS

Per risolvere i problemi con i nodi Amazon EKS, è disponibile uno script predefinito sui nodi all'indirizzo. `/etc/eks/log-collector-script/eks-log-collector.sh` È possibile utilizzare lo script per raccogliere i registri diagnostici per i casi di supporto e per la risoluzione dei problemi generali.

Utilizza il comando seguente per eseguire lo script sul nodo:

```
sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

> **Note**
>
> Se lo script non è presente in quella posizione, puoi scaricarlo ed eseguirlo manualmente con il seguente comando:
>
> ```
> curl -O https://amazon-eks.s3.amazonaws.com/support/log-collector-script/linux/
> eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

Lo script raccoglie le seguenti informazioni di diagnostica:

```
$ sudo bash /etc/eks/log-collector-script/eks-log-collector.sh

      This is version 0.7.8. New versions can be found at https://github.com/awslabs/
amazon-eks-ami/blob/main/log-collector-script/

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
...
...

 Done... your bundled logs are located in /var/log/eks_i-EXAMPLE_2025-03-25_0000-
UTC_0.7.8.tar.gz
```

Le informazioni di diagnostica vengono raccolte e archiviate in:

```
/var/log/eks_i-EXAMPLE_2025-03-25_0000-UTC_0.7.8.tar.gz
```

Per recuperare il pacchetto di log per i nodi Bottlerocket, consulta Bottlerocket Logs per maggiori dettagli.

## Rete runtime container non pronta

Potresti visualizzare un errore `Container runtime network not ready` ed errori di autorizzazione analoghi al seguente:

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

Questo può verificarsi per uno dei seguenti motivi:

1. Non disponi di `aws-auth ConfigMap` sul tuo cluster oppure non include voci per il ruolo IAM con cui hai configurato i tuoi nodi.

   Per risolvere il problema, visualizza le voci esistenti `ConfigMap` *`my-cluster`* sostituendo il comando seguente con il nome del cluster e quindi eseguendo il comando modificato:. `eksctl get iamidentitymapping --cluster` *`my-cluster`* Se ricevi un messaggio di errore dal comando, è possibile che il cluster non disponga di `aws-auth ConfigMap`. Il comando seguente aggiunge una voce a `ConfigMap`. Inoltre, se `ConfigMap` non esiste, è creata dal comando. Sostituiscilo *`111122223333`* con l'ID dell' AWS account per il ruolo IAM e *`myAmazonEKSNodeRole`* con il nome del ruolo del tuo nodo.

   ```
   eksctl create iamidentitymapping --cluster my-cluster \
       --arn arn:aws: iam::111122223333:role/myAmazonEKSNodeRole --group
    system:bootstrappers,system:nodes \
       --username system:node:{{EC2PrivateDNSName}}
   ```

L'ARN del ruolo specificato non può includere un percorso diverso da `/`. Ad esempio, se il nome del tuo ruolo è `development/apps/my-role`, devi cambiarlo in `my-role` quando specifichi l'ARN del ruolo. Assicurati di specificare l'ARN del ruolo IAM del nodo (non l'ARN del profilo dell'istanza).

2. I nodi autogestiti si trovano in un cluster con una versione della piattaforma che corrisponde a quella minima elencata nei prerequisiti dell'argomento Concedi agli utenti IAM l'accesso a Kubernetes con le voci di accesso EKS, tuttavia una voce non è elencata in `aws-auth` `ConfigMap` (consulta l'elemento precedente) per il ruolo IAM del nodo o non esiste una voce di accesso per il ruolo. Per risolvere il problema, visualizza le voci di accesso esistenti *`my-cluster`* sostituendo il comando seguente con il nome del cluster e quindi eseguendo il comando modificato:`aws eks list-access-entries --cluster-name` *`my-cluster`* . Il comando seguente aggiunge una voce di accesso per il ruolo IAM del nodo. Sostituiscilo *`111122223333`* con l'ID dell' AWS account per il ruolo IAM e *`myAmazonEKSNodeRole`* con il nome del ruolo del tuo nodo. Se hai un nodo Windows, sostituiscilo *`EC2_LINUX`* con`EC2_Windows`. Assicurati di specificare l'ARN del ruolo IAM del nodo (non l'ARN del profilo dell'istanza).

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/myAmazonEKSNodeRole --type EC2_LINUX
```

## Timeout dell'handshake TLS

Quando un nodo non è in grado di stabilire una connessione all'endpoint del server API pubblico, è possibile che venga visualizzato un errore analogo al seguente.

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

Il processo `kubelet` riprenderà continuamente e testerà l'endpoint del server API. L'errore può verificarsi anche temporaneamente durante qualsiasi procedura che esegue un aggiornamento in sequenza del cluster nel piano di controllo, ad esempio una modifica della configurazione o un aggiornamento della versione.

Per risolvere il problema, controlla la tabella di routing e i gruppi di sicurezza per assicurarti che il traffico proveniente dai nodi possa raggiungere l'endpoint pubblico.

## InvalidClientTokenId

Se stai utilizzando i ruoli IAM per gli account di servizio di un Pod o DaemonSet distribuiti in un cluster in una AWS regione della Cina e non hai impostato la variabile di `AWS_DEFAULT_REGION` ambiente nelle specifiche, il Pod DaemonSet potrebbe ricevere il seguente errore:

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

Per risolvere il problema, devi aggiungere la variabile di `AWS_DEFAULT_REGION` ambiente al tuo Pod o alla tua DaemonSet specifica, come mostrato nel seguente esempio di specifica del Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

# Prima di poter aggiornare il piano di controllo, i gruppi di nodi devono corrispondere alla versione di Kubernetes

Prima di aggiornare un piano di controllo a una nuova versione di Kubernetes, la versione secondaria dei nodi gestiti e Fargate nel cluster deve essere la stessa della versione corrente del piano di controllo. L'API `update-cluster-version` di Amazon EKS rifiuta le richieste fino a quando non si aggiornano tutti i nodi gestiti Amazon EKS alla versione corrente del cluster. Amazon EKS fornisce API per aggiornare i nodi gestiti. Per informazioni sull'aggiornamento delle versioni di Kubernetes del gruppo di nodi gestiti, consulta the section called "Aggiorna". Per aggiornare la versione di un nodo Fargate, elimina il pod rappresentato dal nodo e implementalo nuovamente dopo aver aggiornato il piano di controllo. Per ulteriori informazioni, consulta the section called "Aggiornamento della versione di Kubernetes".

# Quando si avviano molti nodi, si verificano errori Too Many Requests

Se si avviano più nodi contemporaneamente, si potrebbe visualizzare il messaggio di errore `Too Many Requests` nei registri di esecuzione dei dati utente di Amazon EC2. Ciò può verificarsi perché il piano di controllo è sovraccarico di chiamate `describeCluster`. Il sovraccarico si traduce in limitazione (della larghezza di banda della rete), nodi che non eseguono lo script bootstrap e nodo che non vengono aggiunti al cluster.

Assicurarti che gli argomenti `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` siano trasferiti allo script di bootstrap del nodo. Quando includi questi argomenti, non è necessario che lo script bootstrap crei una chiamata `describeCluster`. Ciò contribuisce a impedire che il piano di controllo si sovraccarichi. Per ulteriori informazioni, consulta the section called "Specifica di un'AMI".

# Risposta agli errori HTTP 401 Unauthorized (Autorizzazione negata) per le richieste del server API Kubernetes

Questi errori sono visualizzati se il token dell'account di servizio di un pod è scaduto in un cluster.

Il server API Kubernetes del cluster Amazon EKS rifiuta le richieste con token più vecchi di 90 giorni. Nelle versioni Kubernetes precedenti, i token non avevano una scadenza. Di conseguenza, i client che si basano su questi token devono aggiornarli entro un'ora. Per impedire al server API Kubernetes di rifiutare la richiesta a causa di un token non valido, la versione dell'SDK client Kubernetes utilizzata dal carico di lavoro deve essere uguale o successiva alle versioni riportate di seguito:

- Go versione `0.15.7` e successive
- Python versione `12.0.0` e successive
- Java versione `9.0.0` e successive
- JavaScript versione `0.10.3` e successive
- Ramo `master` di Ruby
- Haskell versione `0.3.0.0`
- Versione C# `7.0.5` e successive

È possibile identificare tutti i pod esistenti nel cluster che utilizzano token obsoleti. Per ulteriori informazioni, consulta the section called "Token dell'account di servizio".

## La versione della piattaforma Amazon EKS è più avanti di due versioni rispetto all'attuale versione della piattaforma

Ciò può accadere quando Amazon EKS non è in grado di aggiornare automaticamente la versione della piattaforma del cluster. Sebbene ci siano molte cause, di seguito riportiamo quelle più comuni. Se uno di questi problemi si applica al tuo cluster, potrebbe ancora funzionare, ma la sua versione della piattaforma non sarà aggiornata da Amazon EKS.

Problema

Il ruolo IAM del cluster è stato eliminato: questo ruolo è stato specificato al momento della creazione del cluster. Puoi visualizzare quale ruolo è stato specificato con il seguente comando. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

Di seguito viene riportato un output di esempio.

```
eksClusterRole
```

Soluzione

Crea un nuovo ruolo IAM del cluster con lo stesso nome.

Problema

Una sottorete specificata durante la creazione del cluster è stata eliminata: le sottoreti da utilizzare con il cluster sono state specificate durante la creazione del cluster. Puoi visualizzare le sottoreti specificate con il seguente comando. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

Di seguito viene riportato un output di esempio.

```
[
```

```
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

Soluzione

Verifica se gli ID di sottorete esistono nel tuo account.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

Di seguito viene riportato un output di esempio.

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

Nel caso in cui gli ID di sottorete restituiti nell'output non corrispondano agli ID di sottorete specificati al momento della creazione del cluster, se desideri che Amazon EKS aggiorni il cluster, devi modificare le sottoreti utilizzate dal cluster. Questo perché se hai specificato più di due sottoreti al momento della creazione del cluster, Amazon EKS seleziona in modo casuale le sottoreti specificate in cui creare nuove interfacce di rete elastiche. Queste interfacce di rete consentono al piano di controllo di comunicare con i nodi. Amazon EKS non aggiornerà il cluster se la sottorete selezionata non esiste. Non hai alcun controllo su quale delle sottoreti specificate al momento della creazione del cluster Amazon EKS sceglie per creare una nuova interfaccia di rete.

Quando avvii un aggiornamento della versione di Kubernetes per il cluster, l'aggiornamento può non riuscire per lo stesso motivo.

Problema

Un gruppo di sicurezza specificato durante la creazione del cluster è stato eliminato: se durante la creazione del cluster hai specificato dei gruppi di sicurezza, puoi visualizzarne gli ID con il seguente comando. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

Di seguito viene riportato un output di esempio.

```
[
    "sg-EXAMPLE1"
]
```

Se `[]` è restituito, non sono stati specificati gruppi di sicurezza al momento della creazione del cluster e il problema non è dato dalla mancanza di un gruppo di sicurezza. Se vengono restituiti i gruppi di sicurezza, conferma che i gruppi di sicurezza sono presenti nel tuo account.

Soluzione

Verifica se questi gruppi di sicurezza esistono nel tuo account.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

Di seguito viene riportato un output di esempio.

```
[
"sg-EXAMPLE2"
]
```

Se desideri che Amazon EKS aggiorni il cluster, nel caso in cui gli ID dei gruppi di sicurezza restituiti nell'output non corrispondano agli ID dei gruppi di sicurezza specificati al momento della creazione del cluster, devi modificare i gruppi di sicurezza utilizzati dal cluster. Amazon EKS non aggiornerà un cluster se gli ID dei gruppi di sicurezza specificati al momento della creazione del cluster non esistono.

Quando avvii un aggiornamento della versione di Kubernetes per il cluster, l'aggiornamento può non riuscire per lo stesso motivo.

- Non hai almeno sei (anche se ne consigliamo 16) indirizzi IP disponibili in ciascuna delle sottoreti specificate al momento della creazione del cluster. Se non disponi di un numero sufficiente di indirizzi IP disponibili nella sottorete, dovrai liberare gli indirizzi IP nella sottorete oppure modificare le sottoreti utilizzate dal cluster in modo da utilizzare le sottoreti con un numero sufficiente di indirizzi IP disponibili.

- La crittografia dei segreti è stata abilitata al momento della creazione del cluster e la chiave AWS KMS specificata è stata eliminata. Se desideri che Amazon EKS aggiorni il cluster, devi crearne uno nuovo

# Domande frequenti sull'integrità del cluster e codici di errore con percorsi di risoluzione

Amazon EKS rileva problemi con i cluster EKS e l'infrastruttura del cluster e li archivia nell'oggetto integrità della risorsa del cluster EKS. Grazie alle informazioni sull'integrità del cluster è possibile rilevare, risolvere e affrontare più rapidamente eventuali problemi. Ciò consente di creare ambienti applicativi più sicuri e aggiornati. Inoltre, è possibile che tu non possa effettuare l'aggiornamento a versioni più recenti di Kubernetes o che Amazon EKS non sia in grado di installare aggiornamenti di sicurezza su un cluster degradato a causa di problemi relativi all'infrastruttura necessaria o alla configurazione del cluster. Amazon EKS può impiegare tre ore per rilevare problemi o verificare che un problema è stato risolto.

L'integrità di un cluster Amazon EKS rappresenta una responsabilità condivisa tra Amazon EKS e i suoi utenti. Sei responsabile dell'infrastruttura indispensabile per i ruoli IAM e le sottoreti Amazon VPC, nonché di altre infrastrutture necessarie, che devono essere fornite anticipatamente. Amazon EKS rileva le modifiche nella configurazione di questa infrastruttura e del cluster.

Per accedere all'integrità del cluster nella console Amazon EKS, è necessario trovare una tabella denominata Problemi di integrità nella scheda Problemi di integrità del cluster nella dashboard di osservabilità a cui si accede dalla pagina dei dettagli del cluster Amazon EKS. Questi dati saranno disponibili anche richiamando l'`DescribeCluster`azione nell'API EKS, ad esempio dall'interfaccia a riga di AWS comando.

Qual è il vantaggio di utilizzare questa funzionalità?

> Otterrai una maggiore visibilità sullo stato del tuo cluster Amazon EKS, diagnosticherai e risolverai rapidamente eventuali problemi, senza dover dedicare tempo al debug o all'apertura AWS di casi di supporto. Ad esempio: hai eliminato accidentalmente una sottorete per il cluster Amazon EKS, Amazon EKS non sarà in grado di creare interfacce di rete tra account e comandi CLI AWS Kubernetes come exec o logs. `kubectl` `kubectl` Queste operazioni non riusciranno e restituiranno l'errore: `Error from server: error dialing backend: remote error: tls: internal error.` Ora sarà visualizzato un problema di integrità di Amazon EKS con il

seguente messaggio: `subnet-da60e280 was deleted: could not create network interface`.

In che modo questa funzionalità è correlata o funziona con altri servizi? AWS

I ruoli IAM e le sottoreti Amazon VPC sono due esempi di elementi dell'infrastrutture indispensabile per i quali l'integrità del cluster può rilevare problemi. Se tali risorse non sono configurate correttamente, questa funzionalità restituirà informazioni dettagliate.

Un cluster con problemi di salute comporta dei costi?

Sì, ogni cluster Amazon EKS viene fatturato ai prezzi standard di Amazon EKS. La funzionalità integrità del cluster è disponibile senza costi aggiuntivi.

Questa funzionalità funziona con i cluster Amazon EKS su AWS Outposts?

Sì, vengono rilevati problemi relativi ai cluster EKS nel AWS cloud, inclusi i cluster estesi su AWS Outposts e i cluster locali su Outposts. AWS L'integrità del cluster non rileva problemi con Amazon EKS Anywhere o Amazon EKS Distro (). EKS-D

Posso ricevere una notifica quando vengono rilevati nuovi problemi?

Sì. AWS invia un'e-mail e una notifica del Personal Health Dashboard quando vengono rilevati nuovi problemi di integrità del cluster.

La console mi avvisa in caso di problemi di salute?

Sì, qualsiasi cluster con problemi di integrità presenterà un banner di avviso nella parte superiore della console.

Le prime due colonne sono quelle necessarie per i valori di risposta dell'API. Il terzo campo dell' ClusterIssueoggetto Health è ResourceIDS, il cui rendimento dipende dal tipo di problema.

| Codice | Messaggio | ResourceIds | Cluster recuperabile? |
|---|---|---|---|
| SUBNET_NOT_FOUND | Non è stato possibile trovare una o più sottoreti attualmente associate al tuo cluster. Effettua una chiamata all'API update-cluster-config | ID sottorete | Sì |

| Codice | Messaggio | ResourceIds | Cluster recuperabile? |
|---|---|---|---|
| | di Amazon EKS per aggiornare le sottoreti. | | |
| SECURITY_GROUP_NOT_FOUND | Non è stato possibile trovare uno o più gruppi di sicurezza attualmente associati al tuo cluster. Chiama l'API update-cluster-config di Amazon EKS per aggiornare i gruppi di sicurezza. | ID dei gruppi di sicurezza | Sì |
| IP_NOT_AVAILABLE | Una o più sottoreti associate al cluster non dispone di indirizzi IP disponibili sufficienti per consentire ad Amazon EKS di eseguire operazioni di gestione dei cluster. Libera indirizzi nelle sottoreti o associa diverse sottoreti al cluster utilizzando l'API update-cluster-config di Amazon EKS. | ID sottorete | Sì |
| VPC_NOT_FOUND | Non è stato possibile trovare il VPC associato al cluster. È necessario eliminare e ricreare il cluster. | ID VPC | No |

| Codice | Messaggio | ResourceIds | Cluster recuperabile? |
| --- | --- | --- | --- |
| ASSUME_ROLE_ACCESS_DENIED | Il tuo cluster non utilizza il ruolo collegato al servizio Amazon EKS. Non è stato possibile assumere il ruolo associato al tuo cluster per eseguire le operazioni di gestione di Amazon EKS richieste. Verifica che il ruolo esista e disponga della policy di attendibilità richiesta. | Ruolo IAM del cluster | Sì |
| PERMISSION_ACCESS_DENIED | Il tuo cluster non utilizza il ruolo collegato al servizio Amazon EKS. Il ruolo associato al tuo cluster non concede autorizzazioni sufficienti ad Amazon EKS per eseguire le operazioni di gestione richieste. Verifica le policy associate al ruolo del cluster e controlla se vengono applicate policy di negazione separate. | Ruolo IAM del cluster | Sì |

| Codice | Messaggio | ResourceIds | Cluster recuperabile? |
|---|---|---|---|
| ASSUME_ROLE_ACCESS_DENIED_USING_SLR | Non è stato possibile assumere il ruolo collegato al servizio di gestione del cluster Amazon EKS. Verifica che il ruolo esista e disponga della policy di attendibilità richiesta. | Il ruolo collegato al servizio di Amazon EKS | Sì |
| PERMISSION_ACCESS_DENIED_USING_SLR | Il ruolo collegato al servizio di gestione del cluster Amazon EKS non concede autorizzazioni sufficienti ad Amazon EKS per eseguire le operazioni di gestione richieste. Verifica le policy associate al ruolo del cluster e controlla se vengono applicate policy di negazione separate. | Il ruolo collegato al servizio di Amazon EKS | Sì |
| OPT_IN_REQUIRED | Il tuo account non dispone di un abbonamento al servizio Amazon EC2. Aggiorna gli abbonamenti del tuo account nella pagina delle impostazioni dell'account. | N/A | Sì |

| Codice | Messaggio | ResourceIds | Cluster recuperabile? |
|---|---|---|---|
| STS_REGIONAL_ENDPOINT_DISABLED | L'endpoint regionale STS è disabilitato. Abilita l'endpoint per Amazon EKS per eseguire le operazioni di gestione dei cluster richieste. | N/A | Sì |
| KMS_KEY_DISABLED | La chiave AWS KMS associata al cluster è disabilitata. Re-enable la chiave per ripristinare il cluster. | L'ARN della chiave KMS | Sì |
| KMS_KEY_NOT_FOUND | Non siamo riusciti a trovare la chiave AWS KMS associata al tuo cluster. È necessario eliminare e ricreare il cluster. | L'ARN della chiave KMS | No |
| KMS_GRANT_REVOKED | Le sovvenzioni per la chiave AWS KMS associata al cluster vengono revocate. È necessario eliminare e ricreare il cluster. | L'ARN della chiave KMS | No |

| Codice | Messaggio | ResourceIds | Cluster recuperabile? |
| --- | --- | --- | --- |
| ETCD_DB_SIZE_EXCEEDED | Il tuo cluster Amazon EKS ha superato il limite di dimensione del database etcd. etcd è un data store chiave-valore che viene eseguito nel piano di controllo di Kubernetes e mantiene la configurazione e lo stato del cluster. Per evitare che il cluster entri in uno stato degradato, riduci le dimensioni del database etcd rimuovendo gli oggetti Kubernetes non necessari. Per indicazioni sull'identificazione e la pulizia degli oggetti che contribuiscono alle dimensioni del database, consulta Gestire le dimensioni del database etcd sui cluster Amazon EKS. Se continui a riscontrare problemi dopo la pulizia, contatta l'assistenza. AWS | Il Cluster ARN | Sì |

# Estensione delle funzionalità di Amazon EKS con progetti open-source

Questi progetti open source estendono le funzionalità dei cluster Kubernetes in esecuzione all'interno o all'esterno di esso AWS, inclusi i cluster gestiti da Amazon EKS.

## Supporto per il software distribuito su EKS

Quando esamini i documenti di Amazon EKS, troverai riferimenti a vari strumenti e software open-source nelle nostre procedure ed esempi. Questi strumenti includono Server delle metriche Kubernetes e Sistema di gestione delle certificazioni.

Tieni presente che qualsiasi software di terze parti o open source che scegli di distribuire non rientra nell'ambito dei tuoi contratti di supporto. AWS Uno dei vantaggi dell'utilizzo di Kubernetes è la community open-source attiva. Consigliamo di collaborare direttamente con le comunità open-source e i responsabili del progetto pertinenti per stabilire canali di supporto appropriati per tali componenti. Per ulteriori informazioni, consulta i progetti graduati e in fase di incubazione associati alla Cloud Native Computing Foundation (CNCF).

L'ecosistema Kubernetes include numerosi progetti e componenti che prevedono diversi livelli di supporto comunitario, tempi di risposta e casi d'uso previsti. Quando implementi queste tecnologie insieme a EKS, assicurati di comprendere la matrice di supporto per ciascun componente.

AWS mantiene i componenti open source che integriamo nel piano di controllo EKS. Ciò include la nostra pipeline di sicurezza completa che copre la verifica delle build, la scansione delle vulnerabilità, i test di convalida e la gestione delle patch per tutte le immagini e i file binari dei container che distribuiamo. Ad esempio, AWS è responsabile del Kubernetes API Server. Il server API Kubernetes è coperto dall'accordo sul livello di servizio di Amazon EKS. È possibile utilizzare il tuo piano di supporto dei servizi web di Amazon per risolvere problemi con il server API Kubernetes o ottenere indicazioni generali.

È necessario esaminare attentamente il supporto offerto per vari Amazon EKS. Add-ons AWS i componenti aggiuntivi sono l'unico tipo di componente aggiuntivo Amazon EKS completamente supportato da. AWS AWS I componenti aggiuntivi di Marketplace sono supportati principalmente dai partner. AWS I componenti aggiuntivi della community ricevono supporto di base per il ciclo di vita da. AWS Per ulteriori informazioni, consulta Supporto dei componenti aggiuntivi.

Ogni componente aggiuntivo EKS, indipendentemente dal tipo, riceve il supporto di base per il ciclo di vita da EKS, inclusi i componenti aggiuntivi di Marketplace. Il supporto di base per il ciclo di vita include l'installazione e la disinstallazione del componente aggiuntivo. Per ulteriori informazioni sui tipi di Amazon EKS Add-ons disponibili e sui livelli di supporto associati, consulta Ambito di supporto per i componenti aggiuntivi di Amazon EKS. Per visualizzare i componenti aggiuntivi completamente supportati da AWS, consulta i componenti aggiuntivi di Amazon Web Services.

- Per ulteriori informazioni sulle nostre pratiche di sicurezza e sui limiti del supporto, consulta Sicurezza in Amazon EKS.
- Per ulteriori informazioni sui componenti aggiuntivi della community e del AWS marketplace disponibili tramite Amazon EKS Add-ons, consulta l' Add-ons assistenza ??? EKS.

# Strumenti di gestione

Strumenti di gestione correlati per cluster Amazon EKS e Kubernetes.

## eksctl

`eksctl` è un semplice strumento CLI per la creazione di cluster su Amazon EKS.

- URL progetto
- Documentazione progetto
- AWS blog open source: eksctl: cluster Amazon EKS con un comando

## AWS Controller per Kubernetes

Con AWS Controllers for Kubernetes, puoi creare e gestire risorse direttamente dal tuo cluster Kubernetes. AWS

Disponibile nelle funzionalità EKS.

- URL progetto
- AWS blog open source: operatore AWS di servizi per Kubernetes ora disponibile

## kro (Kube Resource Orchestrator)

kro consente di creare API Kubernetes personalizzate che compongono più risorse in astrazioni di livello superiore. I team della piattaforma possono definire modelli riutilizzabili con guardrail, mentre i team applicativi utilizzano API semplici e di alto livello per il provisioning e la gestione delle risorse.

Disponibile in EKS Capabilities. ???

- URL progetto
- Documentazione progetto

## Argo CD

Argo CD è uno strumento dichiarativo e di distribuzione GitOps continua per Kubernetes. Monitora continuamente i tuoi repository Git e sincronizza automaticamente le modifiche ai tuoi cluster.

Disponibile in EKS Capabilities.

- URL progetto
- Documentazione progetto

## CD Flux

Flux è uno strumento utile per gestire la configurazione del cluster quando si usa Git. Utilizza un operatore nel cluster per attivare le distribuzioni all'interno di Kubernetes. Per ulteriori informazioni sugli operatori, vedere OperatorHub.io su GitHub.

- URL progetto
- Documentazione progetto

## CDK per Kubernetes

Con il CDK per Kubernetes (cdk8s), è possibile definire app e componenti Kubernetes utilizzando linguaggi di programmazione familiari. Le app cdk8s sintetizzano in manifesti Kubernetes standard, che possono essere applicati a qualsiasi cluster Kubernetes.

- URL progetto

- Documentazione progetto
- AWS container blog: Presentazione di cdk8s+: Intent-driven API per oggetti Kubernetes

# Rete

Progetti di rete correlati per cluster Amazon EKS e Kubernetes.

## Plug-in CNI di Amazon VPC per Kubernetes

Amazon EKS; supporta reti VPC native tramite il plug-in CNI di Amazon VPC per Kubernetes. Il plugin assegna un indirizzo IP dal VPC a ciascun pod.

- URL progetto
- Documentazione progetto

## AWS Load Balancer Controller per Kubernetes

Il AWS Load Balancer Controller aiuta a gestire AWS Elastic Load Balancer per un cluster Kubernetes. Soddisfa le risorse di Kubernetes Ingress mediante il provisioning di Application Load Balancer. AWS Soddisfa le risorse del servizio Kubernetes mediante il provisioning di Network Load Balancer. AWS

- URL progetto
- Documentazione progetto

## ExternalDNS

ExternalDNS sincronizza i servizi e gli ingressi Kubernetes esposti con i provider DNS, tra cui Amazon Route 53 e Service Discovery. AWS

- URL progetto
- Documentazione progetto

# Machine learning

Progetti di machine learning correlati per cluster Amazon EKS e Kubernetes.

### Kubeflow

Un kit di strumenti di machine learning per Kubernetes.

- URL progetto
- Documentazione progetto
- AWS blog open source: Kubeflow su Amazon EKS

## Auto Scaling

Progetti di scalabilità automatica correlati per cluster Amazon EKS e Kubernetes.

### Cluster Autoscaler

Cluster Autoscaler è uno strumento che consente di regolare automaticamente le dimensioni del cluster Kubernetes in base alle richieste di CPU e memoria.

- URL progetto
- Documentazione progetto
- Workshop Amazon EKS: Cluster Autoscaler

### Karpenter

Karpenter è un Node Autoscaler di Kubernetes progettato per flessibilità, prestazioni e semplicità.

- URL progetto
- Documentazione progetto
- Workshop Amazon EKS: Karpenter

### Escalator

Escalator è un batch o autoscaler orizzontale ottimizzato per i processi per Kubernetes.

- URL progetto
- Documentazione progetto

# Monitoraggio

Progetti di monitoraggio correlati per cluster Amazon EKS e Kubernetes.

## Prometheus

Prometheus è un kit di strumenti di monitoraggio e avvisi di sistema open source.

- URL progetto
- Documentazione progetto
- Workshop Amazon EKS: https://eksworkshop.com/intermediate/240_monitoring/

# Integrazione continua / distribuzione continua

CI/CD Progetti imperativi correlati per i cluster Amazon EKS e Kubernetes.

## Jenkins X

CI/CD soluzione per applicazioni cloud moderne su cluster Amazon EKS e Kubernetes.

- URL progetto
- Documentazione progetto

# Scopri le nuove funzionalità e le roadmap di Amazon EKS

Per maggiori informazioni sulle nuove funzionalità di Amazon EKS, scorri fino al feed Novità nella pagina Novità in AWS. È possibile anche rivedere la roadmap su GitHub, che consente di conoscere le funzionalità e le priorità in arrivo, in modo da poter pianificare come utilizzare Amazon EKS in futuro. È possibile fornirci un feedback diretto sulle priorità della roadmap.

# Cronologia dei documenti

La tabella riportata di seguito illustra alcuni dei principali aggiornamenti e nuove funzionalità della Guida per l'utente di Amazon EKS. Per ricevere notifiche quando viene inserita una nuova voce in questa tabella, è possibile iscriversi al seguente URL con un lettore RSS:

```
https://docs.aws.amazon.com/eks/latest/userguide/doc-history.rss
```

| Modifica | Descrizione | Data |
| --- | --- | --- |
| AWS aggiornamenti gestiti delle policy | È stata aggiunta `wafv2:ListWebACLs` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per consentire ad Amazon EKS Auto Mode Controller di elencare gli ACL Web AWS WAF durante la riconciliazione delle associazioni di Application Load Balancer. Sono state aggiunte `gateway.eks.amazonaws.com.alb/stack` le chiavi, `gateway.eks.amazonaws.com.alb/resource` `gateway.eks.amazonaws.com.nlb/stack` , e `gateway.eks.amazonaws.com.nlb/resource` tag alla lista consentita per la codifica delle risorse in. `AmazonEKSLoadBalancingPolicy` Ciò consente ad Amazon EKS Auto Mode Controller di gestire AWS le risorse per i | 2 settembre 2026 |

| | | |
|---|---|---|
| | carichi di lavoro Kubernetes Gateway API (ALB e NLB). | |
| Endpoint deterministico relativo alla funzionalità Argo CD | Ha documentato il formato dell'endpoint EKS Capabilit y for Argo CD. Il nome host dell'endpoint è deterministico, quindi è possibile crearlo a partire dal nome della funzionalità, dal nome del cluster, dall'ID AWS dell'acco unt e dalla AWS regione oppure recuperarlo dal campo di. `serverUrl describe-capability` | 24 agosto 2026 |
| AWS PrivateLink per l'endpoin t OIDC del cluster | Amazon EKS ora supporta AWS PrivateLink l'endpoint OIDC del cluster (`com.amazo naws.region-code.o idc-eks` ), che consente di raggiungere l' discovery/JWKS endpoint utilizzato dai ruoli IAM per gli account di servizio (IRSA) privatamente dal tuo VPC senza uscita Internet. | 28 luglio 2026 |

| | | |
|---|---|---|
| Supporto EFA per la capacità statica in modalità EKS Auto | È stato aggiunto il supporto per la configurazione delle interfacce di rete EFA sulle risorse EKS Auto Mode. NodeClass È ora possibile definire staticamente le interfacce di rete all'avvio dell'istanza per carichi di lavoro di formazione e inferenza distribuiti ad alte prestazioni. | 21 luglio 2026 |
| Service-linked aggiornamento del ruolo | Aggiunto `AmazonEKS PodIdentityPolicy` all'elenco delle politiche di accesso in `AmazonEKS ServiceRolePolicy` quanto il ruolo collegato al servizio può associars i e dissociarsi dalle voci di accesso al cluster. Con questa aggiunta, Amazon EKS può gestire le credenziali di identità dei pod per i carichi di lavoro in esecuzione su cluster EKS. | 26 giugno 2026 |
| Routing in uscita dal piano di controllo | È stato aggiunto il supporto per l'uscita del piano di controllo indirizzato dal cliente (`controlPlaneEgress Mode=CUSTOMER_ROUT ED` ), che consente di gestire il percorso di rete dal piano di controllo del cluster alle risorse nel VPC. | 18 giugno 2026 |

| | | |
|---|---|---|
| Aggiornamento dei ruoli collegati al servizio | È stata aggiunta `elasticloadbalancing:SetRulePriorities` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per consentire ad Amazon EKS Auto Mode Controller di impostare le priorità delle regole di ascolto specifiche per ALB. È stata aggiunta `ec2:DescribeIpamPools` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per descrivere il pool ipam | 15 giugno 2026 |
| Kubernetes versione `1.36` | È stata aggiunta la versione `1.36` Kubernetes ad Amazon EKS. | 2 giugno 2026 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | Aggiornato `AmazonEBSCSIDriverEKSClusterScopedPolicy` per autorizzare `ec2:AttachVolume` e `ec2:DetachVolume` sulle istanze contrassegnate con il `eks:eks-cluster-name` tag, oltre ai tag `eks:cluster-name` e `ebs.csi.aws.com/cluster-name` esistenti. Ciò aggiunge la compatibilità con i controller che seguono lo standard di etichettatura dei controller EKS, tra cui Karpenter open source e Amazon EKS Auto Mode. | 28 maggio 2026 |
| AWS aggiornamenti gestiti delle politiche | Sono state aggiunte autorizzazioni per consentire `AmazonEKSComputePolicy` ad Amazon EKS Auto Mode di avviare istanze da Interruptible Capacity Reservations nel tuo account. | 18 maggio 2026 |
| AWS aggiornamenti gestiti delle politiche | È stata aggiunta una nuova politica AWS gestita: `AmazonEKSBlockStoragePolicyV2` | 12 maggio 2026 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | È stata aggiunta una nuova azione API `ec2:DescribeInstanceTypes` per consentire `AmazonEBSCSIDriverPolicy` al driver EBS CSI di recuperare dinamicamente le informazioni sul volume limit/card in fase di esecuzione. | 12 maggio 2026 |
| Aggiornamento dei ruoli collegati al servizio | È stata aggiunta `elasticloadbalancing:DescribeLoadBalancers` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per consentire a Amazon EKS Auto Mode Controller di descrivere LoadBalancer | 27 aprile 2026 |
| Versione Kubernetes `1.32` per cluster locali su Outposts AWS | È stato aggiunto il `1.32` supporto della versione Kubernetes per i cluster Amazon EKS locali su Outposts. AWS | 23 aprile 2026 |
| Aggiornamento dei ruoli collegati al servizio | È stata aggiunta `elasticloadbalancing:DescribeCapacityReservation` l'autorizzazione `AmazonEKSLoadBalancingPolicy` per consentire ad Amazon EKS Auto Mode Controller di descrivere le unità LoadBalancer di capacità | 20 aprile 2026 |

| | | |
|---|---|---|
| Aggiornamento dei ruoli collegati al servizio | Aggiunto `ec2:DescribeInstanceStatus` e `iam:ListInstanceProfiles` autorizzazioni inserite. `AmazonEKSServiceRolePolicy` Queste autorizzazioni consentono ad Amazon EKS Auto Mode di rilevare gli eventi di integrità di EC2 per le istanze gestite e di pulire i profili delle istanze gestite | 20 aprile 2026 |
| AWS aggiornamenti gestiti delle politiche | Sono state aggiunte autorizzazioni per consentire `AmazonEKSComputePolicy` ad Amazon EKS Auto Mode di avviare istanze nei gruppi di collocamento EC2 del tuo account. | 17 aprile 2026 |

| | | |
|---|---|---|
| Nuova politica AWS gestita | Introdotta `AmazonEBSCSIDriverPolicyV2` e `AmazonEBSCSIDriverEKSClusterScopedPolicy` come alternative più restrittive a `AmazonEBSCSIDriverPolicy`. `AmazonEBSCSIDriverPolicyV2` assegna il driver EBS CSI ai volumi e alle istantanee contrassegnati con set to. `ebs.csi.aws.com/cluster` `true` `AmazonEBSCSIDriverEKSClusterScopedPolicy` assegna il driver ai volumi e alle istantanee appartenenti a uno specifico cluster EKS utilizzando il tag. `ebs.csi.aws.com/cluster-name` | 16 aprile 2026 |
| AWS aggiornamenti gestiti delle politiche | Sono state aggiunte autorizzazioni per consentire `AmazonEKSServiceRolePolicy` ad Amazon EKS Auto Mode di avviare istanze nei gruppi di collocamento EC2 del tuo account. | 15 aprile 2026 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | Aggiornato `shield:CreateProtection` e `shield:DeleteProtection` autorizzato all'accesso a `AmazonEKSLoadBalancingPolicy`. Ciò consente ad Amazon EKS Auto Mode Controller di aggiungere una protezione Shield. | 14 aprile 2026 |
| Supporto Warm Pool per gruppi di nodi gestiti | I gruppi di nodi gestiti di Amazon EKS ora supportano i warm pool Amazon EC2 Auto Scaling. I warm pool mantengono un pool di istanze EC2 preinizializzate pronte a unirsi rapidamente al cluster in caso di scalabilità orizzontale, riducendo la latenza per le applicazioni con tempi di avvio lunghi. | 7 aprile 2026 |
| AWS aggiornamenti gestiti delle politiche | `elasticloadbalancing:RegisterTargets` Autorizzazione aggiunta `elasticloadbalancing:ModifyIpPools` e aggiornata in `AmazonEKSLoadBalancingPolicy`. Ciò consente ad Amazon EKS Auto Mode Controller di supportare l'impostazione del pool IPAM per ALB e supportare, rispettivamente, multi-cluster e targetgroupbindings personalizzati | 20 marzo 2026 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | Aggiunti `autoscaling:PutWarmPool` `autoscaling:DeleteWarmPool` , e `autoscaling:DescribeWarmPool` autorizzazioni alla `AWSServiceRoleForAmazonEKSNodegroup` politica. Ciò consente ad Amazon EKS Managed Nodegroups di gestire le risorse del warm pool ASG sottostanti per tutto il ciclo di vita del gruppo di nodi. | 17 febbraio 2026 |
| AWS aggiornamenti gestiti delle politiche | È stata aggiunta `ec2:ModifyNetworkInterfaceAttribute` l'autorizzazione in`AmazonEKSNetworkingPolicy` . Ciò consente ad Amazon EKS Auto Mode Controller di modificare gli attributi dell'interfaccia di rete relativi alle istanze EC2 | 3 febbraio 2026 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | È stato rimosso il requisito del prefisso «eks» nel nome del profilo dell'istanza di destinazione per l'`iam:GetInstanceProfile` autorizzazione all'`AmazonEKSServiceRolePolicy` accesso. Ciò consente ad Amazon EKS Auto Mode di convalidare e utilizzare profili di istanza personalizzati NodeClasses senza richiedere il prefisso di denominazione «eks». | 2 febbraio 2026 |
| Kubernetes versione `1.35` | Sono stati aggiunti il supporto per Kubernetes versione `1.35` per nuovi cluster e aggiornamenti della versione. | 27 gennaio 2026 |
| AWS aggiornamenti gestiti delle politiche | È stata aggiunta `ec2:LockSnapshot` l'autorizzazione `AmazonEBSCSIDriverPolicy` per consentire al driver EBS CSI di bloccare direttamente le istantanee EBS. | 15 gennaio 2026 |

| | | |
|---|---|---|
| Funzionalità di Amazon EKS | Amazon EKS ora fornisce funzionalità di cluster completamente gestite con EKS Capabilities, con supporto per la distribuzione continua dichiarativa con supporto per Argo CD, gestione AWS delle risorse con supporto per AWS Controllers for Kubernetes e composizione e orchestrazione delle risorse Kubernetes con supporto per Kube Resource Orchestrator. | 30 novembre 2025 |
| Nuova politica AWS gestita | Amazon EKS ha rilasciato una nuova policy gestita `AmazonEKSMCPReadOnlyAccess` per abilitare strumenti di sola lettura nel server MCP Amazon EKS per l'osservabilità e la risoluzione dei problemi. Per informazioni, consulta gli aggiornamenti di Amazon EKS alle policy gestite. AWS | 21 novembre 2025 |
| Osservabilità della rete | Amazon EKS offre ora una migliore osservabilità della rete di container con monitoraggio delle prestazioni e visibilità del traffico dei carichi di lavoro. | 19 novembre 2025 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | È stata aggiunta `ec2:CopyVolumes` l'autorizzazione `AmazonEBSCSIDriverPolicy` per consentire al driver EBS CSI di copiare direttamente i volumi EBS. | 17 novembre 2025 |
| AWS aggiornamenti gestiti delle politiche | Aggiunti `ec2:DescribeRouteTables` e `ec2:DescribeNetworkAcls` autorizzazioni a. `AmazonEKSServiceRolePolicy` Ciò consente ad Amazon EKS di rilevare problemi di configurazione con le tabelle di routing VPC e gli ACL di rete per i nodi ibridi come parte delle informazioni sui cluster. | 22 ottobre 2025 |
| Aggiornamento dei ruoli collegati al servizio | È stata aggiunta `ssmmessages:OpenDataChannel` l'autorizzazione a `AmazonEKSConnectorServiceRolePolicy` | 15 ottobre 2025 |
| Kubernetes versione `1.34` | Sono stati aggiunti il supporto per Kubernetes versione `1.34` per nuovi cluster e aggiornamenti della versione. | 2 ottobre 2025 |
| Riparazione automatica dei nodi configurabile | Ora Amazon EKS offre un controllo più granulare sul comportamento di riparazione automatica del nodo. | 22 settembre 2025 |

| | | |
|---|---|---|
| Aggiornamento degli approfondimenti sui cluster | Ora è possibile aggiornare manualmente gli approfondimenti sui cluster. | 27 agosto 2025 |
| AWS aggiornamenti gestiti delle politiche | Autorizzazione aggiunta a `AmazonEKSServiceRolePolicy` . Questo ruolo può collegare una nuova policy di accesso `AmazonEKSEventPolicy` . Autorizzazioni limitate per `ec2:DeleteLaunchTemplate` e `ec2:TerminateInstances` . | 26 agosto 2025 |
| Cross-service prevenzione sostitutiva confusa | È stato aggiunto un argomento con un esempio di politica di fiducia che è possibile applicare per la prevenzione dei sostituti Cross-service confusi. Amazon EKS accetta le condizioni `aws:SourceArn` e `aws:SourceAccount` e la policy di attendibilità di un ruolo del cluster EKS. | 19 agosto 2025 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.33.2`, `1.32.6`, `1.31.10` e `1.30.14`. | 30 luglio 2025 |

| | | |
|---|---|---|
| Funzione VPC Multi-NIC CNI per pod multi-homed | Amazon EKS aggiunge i pod multi-homed a CNI VPC. Ora è possibile configura re un carico di lavoro e gli indirizzi IP assegnati da CNI VPC da ogni NIC sull'ista nza EC2 a ciascun pod. L'applicazione può effettuar e connessioni simultanee per utilizzare la larghezza di banda di ciascuna NIC. Ogni interfaccia di rete è configura ta nella stessa sottorete e negli stessi gruppi di sicurezza del nodo. In precedenza, era necessario utilizzare Multus CNI per eseguire più CNI per creare pod multi-homed. La documentazione e i passaggi per eseguire questa operazion e sono stati spostati in. https://docs.aws.amazon.com/eks/latest/userguide/pod-multus.html | 15 luglio 2025 |
| Aggiornamento dei contenuti di risoluzione dei problemi di CNI VPC | La pagina di risoluzione dei problemi relativa alla policy di rete Kubernetes in CNI VPC è stata ampliata. Aggiunte le CRD e le autorizzazioni RBAC, nonché 12 problemi noti. | 30 giugno 2025 |

| | | |
|---|---|---|
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Non sono presenti nuove versioni delle patch di Kubernetes in queste versioni della piattaforma. | 26 giugno 2025 |
| AWS aggiornamenti gestiti delle politiche | Aggiunta l'autorizzazione `ssmmessages:OpenDataChannel` a `AmazonEKSLocalOutpostServiceRolePolicy` . | 26 giugno 2025 |
| AWS aggiornamenti gestiti delle politiche | Sono state aggiunte autorizzazioni per `AmazonEKSServiceRolePolicy` consentire `AmazonEKSComputePolicy` ad Amazon EKS Auto Mode di avviare istanze utilizzando le prenotazioni di On-Demand capacità EC2 nel tuo account. Inoltre, sono state aggiunte autorizzazioni a `AmazonEKSComputePolicy` per consentire ad Amazon EKS di creare il ruolo collegato al servizio EC2 spot per conto dell'utente. | 20 giugno 2025 |
| Aggiornamenti alle policy gestite | Aggiunta la policy `AmazonEKSDashboardConsoleReadOnly` . | 19 giugno 2025 |

| | | |
|---|---|---|
| Aggiornamento della modalità automatica di Amazon EKS a `NodeClass` | Il modello `NodeClass` per i nodi della modalità automatica ha aggiunto la configurazione per sottoreti di pod separati. Vengono aggiunte le chiavi facoltative `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` per impostare le sottoreti e i gruppi di sicurezza per i pod. | 13 giugno 2025 |
| Ruoli secondari e multi-account di destinazione con EKS Pod Identity | Amazon EKS aggiunge ruoli IAM di destinazione alle EKS Pod Identity per il concatenamento dei ruoli. # possibile utilizzarlo per assumere automaticamente un ruolo in un altro account, con EKS Pod Identity che ruota le credenziali temporanee. Ogni associazione Pod Identity deve prima disporre di un ruolo IAM da assumere nello stesso account, poi utilizza tale ruolo per assumere il ruolo di destinazione. | 11 giugno 2025 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Non sono presenti nuove versioni delle patch di Kubernetes in queste versioni della piattaforma. | 11 giugno 2025 |

| | | |
|---|---|---|
| Espansione della regione Amazon EKS AWS | Amazon EKS è ora disponibile nella regione Asia Pacifico (Taipei) (`ap-east-2`). AWS | 6 giugno 2025 |
| Controllo degli accessi `IPv6` per endpoint pubblici dual stack per nuovi cluster `IPv6` | Amazon EKS aggiunge intervalli CIDR `IPv6` per controllare l'accesso all'endpoint pubblico del cluster per nuovi cluster `IPv6`. In precedenza, era possibile aggiungere solo intervalli CIDR `IPv4` per consentire il traffico verso l'endpoint del cluster pubblico, anche per gli endpoint del cluster dual-stack. | 5 giugno 2025 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.32.5`, `1.31.9` e `1.30.13`. | 30 maggio 2025 |
| Nuovi approfondimenti sui cluster per EKS Hybrid Nodes | Amazon EKS aggiunge nuovi approfondimenti sui cluster che verificano la configurazione dei nodi ibridi. Questi controlli approfonditi avviseranno dei problemi relativi ai nodi e ai pod on-premises e alla configurazione di rete remota del cluster. | 29 maggio 2025 |

| | | |
|---|---|---|
| Kubernetes versione 1.33 | Sono stati aggiunti il supporto per Kubernetes versione 1.33 per nuovi cluster e aggiornamenti della versione. | 29 maggio 2025 |
| Add-on supporto per il driver Amazon FSx CSI | Ora puoi utilizzare la console di AWS gestione, l'interfaccia a riga di comando e l'API per gestire il driver Amazon FSx AWS CSI. | 23 maggio 2025 |
| Modifica degli scraper Prometheus nella console | Ora è possibile modificare gli scraper del Servizio gestito da Amazon per Prometheus nella console Amazon EKS. | 22 maggio 2025 |
| Aggiornamenti alle policy gestite | Aggiunta la policy AmazonEKSDashboardServiceRolePolicy . | 21 maggio 2025 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Non sono presenti nuove versioni delle patch di Kubernetes in queste versioni della piattaforma. | 16 maggio 2025 |
| Nuove pagine per le prestazioni di Amazon FSx per Lustre | Aggiunti nuovi argomenti con dettagli sull'ottimizzazione delle prestazioni di Amazon FSx per Lustre. | 2 maggio 2025 |

| | | |
|---|---|---|
| Aggiornamento della modalità automatica di Amazon EKS a `NodeClass` | Il modello `NodeClass` per i nodi della modalità automatic a ha aggiunto la configura zione per i proxy di inoltro di rete. Ciò aggiunge la chiave facoltativa `advancedN etworking` per configurare il proxy HTTPS. | 30 aprile 2025 |
| Bottlerocket per nodi ibridi | Bottlerocket è ora disponibile per EKS Hybrid Nodes. | 29 aprile 2025 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzion i della sicurezza e miglioram enti. Non sono presenti nuove versioni delle patch di Kubernetes in queste versioni della piattaforma. | 29 aprile 2025 |
| Nuove pagine con concetti relativi alla rete ibrida | Aggiunte pagine con concetti relativi a EKS Hybrid Nodes. Vengono trattate nel dettaglio le reti on-premises e cloud con diagrammi. | 18 aprile 2025 |
| AWS aggiornamenti gestiti delle policy | Sono state aggiunte autorizzazioni per consentir e `AmazonEKSClusterPo licy` ad Amazon EKS di ripulire le interfacce di rete elastiche create dal VPC CNI. Ciò è necessario affinché EKS possa ripulire le interfacce di rete elastiche che rimangono se CNI VPC si chiude inaspettatamente. | 16 aprile 2025 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle policy | Sono state aggiunte autorizzazioni per consentire `AmazonEKSServiceRolePolicy` ai AI/ML clienti EKS di aggiungere regole di uscita al gruppo di sicurezza predefinito di EKS Cluster. | 14 aprile 2025 |
| La versione 1.31 di Kubernetes è ora disponibile per i cluster locali su Outposts AWS | Ora puoi creare un cluster locale Amazon EKS su un Outposts utilizzando la versione 1.31 di Kubernetes AWS . | 8 aprile 2025 |
| Integrità dei nodi per EKS Hybrid Nodes | È possibile utilizzare `eks-node-monitoring-agent` su nodi ibridi, a partire dalla versione `1.2.0-eksbuild.1` . Eseguire `eks-node-monitoring-agent` come componente aggiuntivo di Amazon EKS per rilevare e mostrare problemi di integrità. | 31 marzo 2025 |

| | | |
|---|---|---|
| EKS Hybrid Nodes per cluster esistenti | Ora è possibile aggiunger e, modificare o rimuovere la configurazione dei nodi ibridi dei cluster esistenti. In precedenza, era possibile aggiungere la configura zione dei nodi ibridi ai nuovi cluster solo al momento della creazione. Con Amazon EKS Hybrid Nodes, puoi utilizzare la tua infrastruttura locale ed edge come nodi nei cluster Amazon EKS. AWS gestisce il piano AWS di controllo Kubernetes ospitato del cluster Amazon EKS e tu gestisci i nodi ibridi che vengono eseguiti nei tuoi ambienti locali o periferici. | 31 marzo 2025 |
| Rollback: prevenire gli aggiornamenti accidentali con gli approfondimenti sui cluster | Amazon EKS ha temporane amente eseguito il rollback di una funzionalità che richiedev a l'utilizzo di un flag `--force` per aggiornare il cluster in caso di determinati problemi di approfondimenti sul cluster. Per ulteriori informazioni, consulta Temporary rollback of enforcing upgrade insights on Update Cluster version on. GitHub | 28 marzo 2025 |
| AMI Bottlerocket FIPS | Le AMI Bottlerocket FIPS sono ora disponibili in gruppi di nodi gestiti standard. | 27 marzo 2025 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle policy | Aggiunte autorizzazioni a `AmazonEKSServiceRolePolicy` per consentire ad Amazon EKS di terminare le istanze EC2 create dalla modalità automatica. | 28 febbraio 2025 |
| Strategie di aggiornamento per i gruppi di nodi gestiti | Ora è possibile utilizzare strategie di aggiornamento per configurare il processo di aggiornamento delle versioni per i gruppi di nodi gestiti. Ciò introduce la strategia di aggiornamento minima per terminare i nodi prima di crearne di nuovi, utile in ambienti con capacità limitata. La strategia di aggiornamento predefinita mantiene il comportamento esistente. | 27 gennaio 2025 |
| Kubernetes versione `1.32` | Sono stati aggiunti il supporto per Kubernetes versione `1.32` per nuovi cluster e aggiornamenti della versione. | 23 gennaio 2025 |
| Espansione AWS della regione Amazon EKS | Amazon EKS è ora disponibile nelle Regioni AWS Asia Pacifico (Thailandia) (`ap-southeast-7`) e Messico (Centrale) (`mx-central-1`). La modalità automatica EKS e gli endpoint VPC per l'API EKS non sono disponibili in nessuna delle due Regioni. | 14 gennaio 2025 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | Aggiunte più autorizzazioni a `AmazonEBSCSIDriverPolicy` per consentire al driver CSI per Amazon EBS di ripristinare tutti gli snapshot, abilitare il ripristino rapido degli snapshot (FSR) sui volumi EBS e modificare i tag sui volumi. | 13 gennaio 2025 |
| AWS aggiornamenti gestiti delle politiche | Aggiunte autorizzazioni a `AmazonEKSLoadBalancingPolicy` . | 26 dicembre 2024 |
| Approfondimenti sui cluster aggiornati | Ora gli approfondimenti di aggiornamento di Amazon EKS avviseranno di più problemi relativi all'integrità del cluster e alla compatibilità delle versioni. Possono rilevare problemi tra diversi componenti di Kubernetes e Amazon EKS, come `kubelet` e `kube-proxy` , nonché i componenti aggiuntivi Amazon EKS. | 20 dicembre 2024 |

| | | |
|---|---|---|
| Agente di monitoraggio e riparazione automatica dei nodi | # possibile utilizzare il nuovo `eks-node-monitoring-agent` come componente aggiuntivo di Amazon EKS per rilevare e mostrare problemi di integrità. # anche possibile abilitare la riparazione automatica dei nodi per sostituire automaticamente i nodi quando vengono rilevati problemi. | 16 dicembre 2024 |
| Amazon EKS Hybrid Nodes | Ora è possibile eseguire il nodo on-premises connesso ai cluster Amazon EKS. Con Amazon EKS Hybrid Nodes, puoi utilizzare la tua infrastruttura locale ed edge come nodi nei cluster Amazon EKS. AWS gestisce il piano AWS di controllo Kubernetes ospitato del cluster Amazon EKS e tu gestisci i nodi ibridi che vengono eseguiti nei tuoi ambienti locali o periferici. | 1 dicembre 2024 |

| | | |
|---|---|---|
| Modalità automatica di Amazon EKS | La modalità automatica di Amazon EKS automatizza completamente la gestione dell'infrastruttura del cluster Kubernetes per calcolo, archiviazione e rete su AWS. Semplifica la gestione di Kubernetes effettuando automaticamente il provisioning dell'infrastruttura, selezionando le istanze di calcolo ottimali, scalando dinamicamente le risorse, ottimizzando continuamente i costi, applicando patch ai sistemi operativi e integrandosi con i servizi di sicurezza. AWS | 1 dicembre 2024 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.31.2`, `1.30.6`, `1.29.10` e `1.28.15`. | 22 novembre 2024 |
| AWS aggiornamenti gestiti delle policy | Aggiornato `AWSServiceRoleForAmazonEKSNodegroup` per compatibilità con le regioni della Cina. | 22 novembre 2024 |
| La versione 1.30 di Kubernetes è ora disponibile per i cluster locali su Outposts AWS | Ora puoi creare un cluster locale Amazon EKS su un Outposts utilizzando la versione 1.30 di Kubernetes AWS . | 21 novembre 2024 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle policy | Politica AWS gestita aggiornata da EKS`AmazonEKSLocalOutpostClusterPolicy`. È stata aggiunta l'`ec2:DescribeAvailabilityZones` autorizzazione in modo che il AWS Cloud Controller Manager sul piano di controllo del cluster possa identificare la zona di disponibilità in cui si trova ciascun nodo. | 21 novembre 2024 |
| AMI Bottlerocket che utilizzano FIPS 140-3 | Sono disponibili AMI Bottlerocket preconfigurate per utilizzare moduli crittografici convalidati FIPS 140-3. Ciò include il modulo crittografico Amazon Linux 2023 Kernel Crypto API e il modulo crittografico AWS-LC. | 20 novembre 2024 |
| AWS aggiornamenti gestiti delle policy | Policy `AWSServiceRoleForAmazonEKSNodegroup` aggiornata per consentire `ec2:RebootInstances` per le istanze create da gruppi di nodi gestiti di Amazon EKS. Limitazione delle autorizzazioni `ec2:CreateTags` per le risorse Amazon EC2. | 20 novembre 2024 |

| | | |
|---|---|---|
| Dashboard di osservabilità | La dashboard di osservabilità consente di rilevare, risolvere e correggere rapidamente i problemi. Nel namespace sono disponibili anche nuove metriche CloudWatch vendute. `AWS/EKS` | 18 novembre 2024 |
| AWS aggiornamenti gestiti delle policy | Politica AWS gestita aggiornata da EKS`AmazonEKS ServiceRolePolicy` . Aggiunte le autorizzazioni per le politiche di accesso EKS, la gestione del bilanciatore del carico e la pulizia automatica delle risorse del cluster. | 16 novembre 2024 |
| Creazione di nuovi ruoli nella console per componenti aggiuntivi che supportano EKS Pod Identity | Sono stati aggiunti nuovi passaggi quando si utilizza la console per creare o aggiornare componenti aggiuntivi che supportano EKS Pod Identity che consentono di generare automaticamente ruoli IAM con il nome, la policy dei ruoli e la policy di attendibilità adeguati per il componente aggiuntivo. | 15 novembre 2024 |
| Gruppi di nodi gestiti nelle zone AWS locali | I gruppi di nodi gestiti possono ora essere creati nelle zone AWS locali. | 15 novembre 2024 |
| Disponibili nuovi parametri | Sono disponibili nuovi parametri nel gruppo API `metrics.eks.amazonaws.com` . | 11 novembre 2024 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle policy | Politica AWS gestita aggiornata da EKS`AmazonEKSComputePolicy`. Autorizzazioni delle risorse aggiornate per l'azione `iam:AddRoleToInstanceProfile`. | 7 novembre 2024 |
| AWS aggiornamenti gestiti delle politiche | EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSComputePolicy` | 1 novembre 2024 |
| AWS aggiornamenti gestiti delle politiche | Aggiunte autorizzazioni a `AmazonEKSClusterPolicy`. Aggiunta l'autorizzazione `ec2:DescribeInstanceTopology` per consentire ad Amazon EKS di collegare informazioni di topologia al nodo come etichette. | 1 novembre 2024 |
| AWS aggiornamenti gestiti delle politiche | EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSBlockStoragePolicy` | 30 ottobre 2024 |
| AWS aggiornamenti gestiti delle politiche | EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSLoadBalancingPolicy` | 30 ottobre 2024 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche | Sono state aggiunte `cloudwatch:PutMetricData` autorizzazioni per consentire `AmazonEKSServiceRolePolicy` ad Amazon EKS di pubblicare metriche su Amazon. CloudWatch | 29 ottobre 2024 |
| AWS aggiornamenti gestiti delle policy | EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSNetworkingPolicy` | 28 ottobre 2024 |
| Dual-stack endpoint per nuovi cluster `IPv6` | Connettersi a nuovi cluster `IPv6` con un endpoint `eks-cluster.` *`region`*`.api.aws` dual stack. Questo endpoint viene restituito quando si descrivono questi cluster. `kubectl` e altri client API Kubernetes in `IPv4`, `IPv6` o ambienti dual stack possono risolvere e connettersi a questi endpoint per cluster pubblici o privati. | 21 ottobre 2024 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle policy | Aggiunte `autoscaling:ResumeProcesses`, `autoscaling:SuspendProcesses` e autorizzazioni associate a `AWSServiceRoleForAmazonEKSNodegroup` nelle Regioni della Cina per l'integrazione con Amazon Application Recovery Controller per EKS. Nessuna modifica in altre Regioni. | 21 ottobre 2024 |
| AMI AL2023 accelerate | Ora puoi utilizzare istanze accelerate `NVIDIA` e AWS Neuron per le AMI basate su AL2023. | 11 ottobre 2024 |
| Nuovo formato di origine | Abbiamo effettuato il passaggio a un nuovo formato di origine con alcune modifiche al layout. Stiamo risolvendo alcuni problemi di formattazione minori e temporanei. | 10 ottobre 2024 |
| AWS aggiornamenti gestiti delle policy | Aggiunte autorizzazioni a `AmazonEKSServicePolicy` e `AmazonEKSServiceRolePolicy`. Aggiunte le autorizzazioni `ec2:GetSecurityGroupsForVpc` e tag associati per consentire a EKS di leggere le informazioni sui gruppi di sicurezza e aggiornare i tag correlati. | 10 ottobre 2024 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche - Nuova politica | EKS ha aggiunto una nuova politica AWS gestita. | 3 ottobre 2024 |
| Kubernetes versione `1.31` | Sono stati aggiunti il supporto per Kubernetes versione `1.31` per nuovi cluster e aggiornamenti della versione. | 24 settembre 2024 |
| AWS aggiornamenti gestiti delle politiche - Aggiornamento di una politica esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 21 agosto 2024 |
| La versione 1.29 di Kubernetes è ora disponibile per i cluster locali su Outposts AWS | Ora puoi creare un cluster locale Amazon EKS su un Outposts utilizzando la versione 1.29 di Kubernetes AWS . | 20 agosto 2024 |
| EKS Pod Identity in (Stati Uniti) AWS GovCloud | Le Amazon EKS Pod Identity associano un ruolo IAM a un account di servizio Kubernetes. Grazie a questa funzionalità, non è più necessario fornire autorizzazioni estese al ruolo IAM del nodo. In questo modo, i pod su quel nodo possono chiamare le AWS API. A differenza dei ruoli IAM per gli account di servizio, le EKS Pod Identity sono completamente interne a EKS; non è necessario un gestore di identità OIDC. | 14 agosto 2024 |

| | | |
|---|---|---|
| Scenario-driven aggiornamenti dei contenuti | Abbiamo rinominato e aggiornato gli argomenti affinché siano più basati sugli argomenti nell'intera guida. | 9 agosto 2024 |
| Dual-stack Endpoint di interfaccia VPC per Amazon EKS | Ora è possibile creare endpoint di interfaccia VPC dual stack per Amazon EKS con indirizzi IP e nomi DNS `IPv4` e `IPv6`. | 7 agosto 2024 |
| Nuovi endpoint dual stack per le API Amazon EKS con indirizzi `IPv6` | L'API EKS per la creazione e la gestione di cluster e gli URL dell'emittente OIDC per i cluster dispongon o di nuovi endpoint dual stack. Il nuovo nome DNS per l'API Amazon EKS è `eks.`*`region`*`.api.aws`, che si risolve in indirizzi `IPv4` e `IPv6`. I nuovi cluster hanno un nuovo URL dell'emittente OIDC dual stack (`oidc-eks.`*`region`*`.api.aws`). | 1° agosto 2024 |
| Blocchi di capacità per gruppi di nodi gestiti | Ora è possibile utilizzare blocchi di capacità per gruppi di nodi gestiti. | 1° luglio 2024 |

| | | |
|---|---|---|
| Abilitazione della raccolta dei parametri di un gruppo Auto Scaling per impostazione predefinita | Nei gruppi di nodi gestiti Amazon EKS, i parametri dei gruppi Amazon EC2 Auto Scaling sono abilitati per impostazione predefinita senza costi aggiuntivi. In precedenza, per abilitare questa funzionalità era necessario eseguire diversi passaggi. | 28 giugno 2024 |
| AWS aggiornamenti gestiti delle policy: aggiornamento a una policy esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 27 giugno 2024 |
| Kubernetes versione `1.26` | La versione `1.26` di Kubernetes si trova ora nel supporto esteso. | 12 giugno 2024 |
| Miglioramenti ai riferimenti informativi sulle AMI | Sono stati apportati miglioramenti ai riferimenti informativi sulle AMI, soprattutto per Bottlerocket. | 12 giugno 2024 |
| Kubernetes versione `1.30` | Sono stati aggiunti il supporto per Kubernetes versione `1.30` per nuovi cluster e aggiornamenti della versione. | 23 maggio 2024 |

| | | |
|---|---|---|
| Dimensionamento automatico di CoreDNS | L'autoscaler di CoreDNS adatterà dinamicamente il numero di repliche dell'implementazione CoreDNS in un cluster EKS in base al numero di nodi e core CPU. Questa funzionalità è disponibile per CoredNS `v1.9` e per la versione della piattaforma più recente di EKS versione `1.25` e successive. | 14 maggio 2024 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.29.4`, `1.28.9` e `1.27.13`. | 14 maggio 2024 |
| CloudWatch Supporto per Container Insights per Windows | Il componente aggiuntivo Amazon CloudWatch Observability Operator ora consente anche di utilizzare Container Insights sui nodi di lavoro Windows del cluster. | 10 aprile 2024 |
| Concetti di Kubernetes | È stato aggiunto un nuovo argomento relativo ai concetti di Kubernetes. | 5 aprile 2024 |

| | | |
|---|---|---|
| Ristrutturazione dei contenuti relativi ad accesso e IAM | Le pagine esistenti relative agli argomenti di accesso e IAM, come mappa di configurazione dell'autenticazione auth config map, voci di accesso, ID pod e IRSA, sono state spostate in una nuova sezione. Revisione del contenuto di panoramica. | 2 aprile 2024 |
| Supporto del sistema operativo Bottlerocket per il driver CSI di Amazon S3 | Il driver CSI di Mountpoint per Amazon S3 è ora compatibile con Bottlerocket. | 13 marzo 2024 |
| AWS aggiornamenti gestiti delle policy: aggiornamento a una policy esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 4 marzo 2024 |
| Amazon Linux 2023 | Amazon Linux 2023 (AL2023) è un nuovo sistema Linux-based operativo progettato per fornire un ambiente sicuro, stabile e ad alte prestazioni per le tue applicazioni cloud. | 29 febbraio 2024 |
| EKS Pod Identity e IRSA supportano i sidecar in Kubernetes `1.29` | In Kubernetes `1.29`, i container sidecar sono disponibili nei cluster Amazon EKS. I container sidecar sono supportati con ruoli IAM per gli account di servizio o EKS Pod Identity. Per ulteriori informazioni sui container sidecar, consultare Sidecar Containers nella documentazione di Kubernetes. | 26 febbraio 2024 |

| | | |
|---|---|---|
| Kubernetes versione `1.29` | Sono stati aggiunti il supporto per Kubernetes versione `1.29` per nuovi cluster e aggiornamenti della versione. | 23 gennaio 2024 |
| Versione completa: supporto esteso di Amazon EKS per le versioni Kubernetes | Il supporto esteso per le versioni Kubernetes consente di utilizzare una versione di Kubernetes specifica per più di 14 mesi. | 16 gennaio 2024 |
| Rilevamento dello stato del cluster Amazon AWS EKS nel cloud | Amazon EKS rileva i problemi nei cluster Amazon EKS e i prerequisiti necessari per l'infrastruttura del cluster in integrità del cluster. Puoi visualizzare i problemi con i tuoi cluster EKS all'interno Console di gestione AWS e all'interno `health` del cluster nell'API EKS. Questi problemi si aggiungono a quelli rilevati e visualizzati dalla console. In precedenza, l'integrità dei cluster era disponibile solo per i cluster locali su AWS Outposts. | 28 dicembre 2023 |
| Approfondimenti sui cluster | Ora è possibile ottenere suggerimenti relativi al cluster sulla base di controlli ricorrenti. | 20 dicembre 2023 |
| Espansione della regione Amazon EKS AWS | Amazon EKS è ora disponibile nella regione Canada occidentale (Calgary) (`ca-west-1` ). AWS | 20 dicembre 2023 |

| | | |
|---|---|---|
| Utilizzando le voci di accesso, ora è possibile concedere ai ruoli e agli utenti IAM l'accesso al cluster | Prima dell'introduzione delle voci di accesso, l'accesso al cluster veniva concesso ai ruoli e agli utenti IAM tramite l'aggiunta di voci a `aws-auth ConfigMap` . Adesso ogni cluster dispone di una modalità di accesso ed è possibile passare all'utilizzo di voci di accesso sulla base della propria pianificazione. Dopo aver cambiato modalità, puoi aggiungere utenti aggiungendo voci di accesso nella AWS CLI e negli AWS CloudFormation SDK. AWS | 18 dicembre 2023 |
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.28.4`, `1.27.8`, `1.26.11` e `1.25.16`. | 12 dicembre 2023 |
| Driver CSI di Mountpoint per Amazon S3 | Ora è possibile installare il driver CSI di Mountpoint per Amazon S3 sui cluster Amazon EKS. | 27 novembre 2023 |

| | | |
|---|---|---|
| Attivazione dei parametri Prometheus durante la creazione di un cluster | In Console di gestione AWS, ora puoi attivare le metriche di Prometheus durante la creazione di un cluster. È anche possibile visualizzare i dettagli dello scraper Prometheus nella scheda Osservabilità. | 26 novembre 2023 |
| Associazioni Amazon EKS Pod Identity | Le Amazon EKS Pod Identity associano un ruolo IAM a un account di servizio Kubernetes. Grazie a questa funzionalità, non è più necessario fornire autorizzazioni estese al ruolo IAM del nodo. In questo modo, i pod su quel nodo possono chiamare le API. AWS A differenza dei ruoli IAM per gli account di servizio, le EKS Pod Identity sono completamente interne a EKS; non è necessario un gestore di identità OIDC. | 26 novembre 2023 |
| AWS aggiornamenti gestiti delle policy - Aggiornamento a una policy esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 26 novembre 2023 |
| Controller di snapshot CSI | Ora puoi installare il controller di snapshot CSI per utilizzarlo con i driver CSI compatibili, come il driver CSI di Amazon EBS. | 17 novembre 2023 |

| | | |
|---|---|---|
| Riscrittura dell'argomento Operatore ADOT | La sezione dedicata al supporto aggiuntivo Amazon EKS per ADOT Operator era superflua con la Distro per la AWS documentazione. OpenTelemetry Abbiamo migrato le informazioni essenziali rimanenti verso quella risorsa per ridurre le informazioni obsolete e incoerenti. | 14 novembre 2023 |
| Supporto per il componente aggiuntivo CoreDNS EKS per i parametri Prometheus | Le versioni `v1.10.1-eksbuild.5` , `v1.9.3-eksbuild.9` e `v1.8.7-eksbuild.8` del componente aggiuntivo EKS per CoreDNS espongono la porta su cui CoreDNS ha pubblicato i parametri nel servizio `kube-dns`. Ciò semplifica l'inclusione dei parametri CoreDNS nei sistemi di monitoraggio. | 10 novembre 2023 |
| Componente aggiuntivo Amazon EKS Observability Operator CloudWatch | È stata aggiunta la pagina Amazon EKS CloudWatch Observability Operator. | 6 novembre 2023 |
| Blocchi di capacità per istanze P5 autogestite nella Regione Stati Uniti orientali (Ohio) | Negli Stati Uniti orientali (Ohio), puoi ora utilizzare blocchi di capacità per istanze P5 autogestite. | 31 ottobre 2023 |

| | | |
|---|---|---|
| I cluster supportano la modifica di sottoreti e gruppi di sicurezza | Puoi aggiornare il cluster per modificare le sottoreti e i gruppi di sicurezza da esso utilizzati. È possibile eseguire l' Console di gestione AWS aggiornamento dalla versione più recente della AWS CLI e da una `eksctl` versione `v0.164.0-rc.0` o successiva. AWS CloudFormation Potrebbe essere necessario eseguire questa operazione per fornire alle sottoreti più indirizzi IP disponibili per aggiornare in modo corretto una versione del cluster. | 24 ottobre 2023 |
| Il ruolo del cluster e il ruolo del gruppo di nodi gestiti supportano le politiche di gestione delle AWS identità e degli accessi gestite dal cliente | È possibile utilizzare una policy IAM personalizzata sul ruolo del cluster, anziché la policy AmazonEKSClusterPolicy AWS gestita. Inoltre, puoi utilizzare una policy IAM personalizzata sul ruolo del nodo in un gruppo di nodi gestiti, anziché la policy AmazonEKSWorkerNodePolicy AWS gestita. Esegui questa operazione per creare una policy con privilegio minimo per soddisfare i rigorosi requisiti di conformità. | 23 ottobre 2023 |

| | | |
|---|---|---|
| Correzione del collegamento per l'installazione di eksctl | Dopo lo spostamento della pagina, correggi il collegamento di installazione per eksctl. | 6 ottobre 2023 |
| Versione di anteprima: supporto esteso di Amazon EKS per le versioni Kubernetes | Il supporto esteso per le versioni Kubernetes consente di utilizzare una versione di Kubernetes specifica per più di 14 mesi. | 4 ottobre 2023 |
| Rimuovi i riferimenti all'integrazione di AWS App Mesh | Le integrazioni di Amazon EKS con AWS App Mesh rimangono disponibili solo per i clienti esistenti di App Mesh. | 29 settembre 2023 |
| Kubernetes versione `1.28` | Sono stati aggiunti il supporto per Kubernetes versione `1.28` per nuovi cluster e aggiornamenti della versione. | 26 settembre 2023 |
| I cluster esistenti supportano l'applicazione della policy di rete di Kubernetes nel plugin CNI di Amazon VPC per Kubernetes | È possibile utilizzare la policy di rete di Kubernetes nei cluster esistenti con il plugin CNI di Amazon VPC per Kubernetes anziché richiedere una soluzione di terze parti. | 15 settembre 2023 |
| Il componente aggiuntivo CoreDNS di Amazon EKS supporta la modifica di PDB | È possibile modificare il `PodDisruptionBudget` del componente aggiuntivo CoreDNS di EKS nelle versioni `v1.9.3-eksbuild.7` e successive e `v1.10.1-eksbuild.4` e successive. | 15 settembre 2023 |

| | | |
|---|---|---|
| Supporto Amazon EKS per sottoreti condivise | Nuova sezione Requisiti e considerazioni condivisi per le sottoreti per creare cluster Amazon EKS in sottoreti condivise. | 7 settembre 2023 |
| Aggiornamenti a Cos'è Amazon EKS? | Sono stati aggiunti nuovi argomenti nelle sezioni Casi d'uso comuni e Architettura. Altri argomenti sono stati aggiornati. | 6 settembre 2023 |
| Applicazione della policy di rete di Kubernetes nel plugin CNI di Amazon VPC per Kubernetes | È possibile utilizzare la policy di rete di Kubernetes con il plugin CNI di Amazon VPC per Kubernetes anziché richiedere una soluzione di terze parti. | 29 agosto 2023 |
| Espansione della AWS regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Israele (Tel Aviv) (`il-central-1` ). | 1° agosto 2023 |
| Archiviazione temporanea configurabile per Fargate | È possibile aumentare la quantità totale di spazio di archiviazione effimero per ciascun pod in esecuzione su Amazon EKS Fargate. | 31 luglio 2023 |
| Add-on supporto per il driver CSI Amazon EFS | Ora puoi utilizzare l' Console di gestione AWS interfacc ia a riga di AWS comando e l'API per gestire il driver CSI di Amazon EFS. | 26 luglio 2023 |
| AWS aggiornamenti gestiti delle policy - Nuova policy | Amazon EKS ha aggiunto una nuova policy AWS gestita. | 26 luglio 2023 |

| | | |
|---|---|---|
| Gli aggiornamenti delle versioni di Kubernetes per 1.27, 1.26, 1.25 e 1.24 sono ora disponibili per i cluster locali su Outposts AWS | Gli aggiornamenti delle versioni di Kubernetes a 1.27.3, 1.26.6, 1.25.11 e 1.24.15 sono ora disponibili per i cluster locali su Outposts AWS | 20 luglio 2023 |
| Supporto dei prefissi IP per nodi Windows | L'assegnazione di prefissi IP ai nodi può consentire di ospitare un numero di gran lunga maggiore di pod sui nodi rispetto a quanti è possibile ospitarne assegnando singoli indirizzi IP secondari ai nodi. | 06 luglio 2023 |
| Driver CSI per Amazon FSx per OpenZFS | Adesso puoi installare il driver CSI di Amazon FSx per OpenZFS sui cluster Amazon EKS. | 30 giugno 2023 |
| I pod sui nodi Linux nei cluster `IPv4` ora possono comunicare con gli endpoint `IPv6`. | Dopo aver assegnato un indirizzo IPv6 al nodo, l'indirizzo `IPv4` dei pod corrisponde all'indirizzo di rete tradotto nell'indirizzo `IPv6` del nodo su cui è in esecuzione. | 19 giugno 2023 |
| Gruppi di nodi AWS GovCloud gestiti da Windows nelle regioni (USA) | Nelle regioni AWS GovCloud (Stati Uniti), i gruppi di nodi gestiti di Amazon EKS possono ora eseguire contenitori Windows. | 30 maggio 2023 |
| Kubernetes versione `1.27` | Sono stati aggiunti il supporto per Kubernetes versione `1.27` per nuovi cluster e aggiornamenti della versione. | 24 maggio 2023 |

| | | |
|---|---|---|
| Kubernetes versione 1.26 | Sono stati aggiunti il supporto per Kubernetes versione 1.26 per nuovi cluster e aggiornamenti della versione. | 11 aprile 2023 |
| gMSA senza dominio | Ora è possibile usare gMSA senza dominio con i pod Windows. | 27 marzo 2023 |
| Espansione AWS della regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Melbourne `ap-southeast-4` ) (). | 10 marzo 2023 |
| Driver CSI di Amazon File Cache | Ora è possibile installare il driver CSI per Amazon File Cache sui cluster Amazon EKS. | 3 marzo 2023 |
| La versione 1.25 di Kubernetes è ora disponibile per i cluster locali su Outposts AWS | Ora è possibile creare un cluster locale Amazon EKS su un Outpost utilizzando le versioni 1.22-1.25 di Kubernetes. | 1 marzo 2023 |
| Kubernetes versione 1.25 | Sono stati aggiunti il supporto per Kubernetes versione 1.25 per nuovi cluster e aggiornamenti della versione. | 22 febbraio 2023 |
| AWS aggiornamenti gestiti delle policy - Aggiornamento a una policy esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 7 febbraio 2023 |

| | | |
|---|---|---|
| Espansione AWS della regione Amazon EKS | Amazon EKS è ora disponibile nelle regioni Asia Pacifico (Hyderabad) (`ap-south-2`), Europa (Zurigo) (`eu-central-2`) ed Europa (Spagna) (`eu-south-2`) AWS . | 6 febbraio 2023 |
| Versioni Kubernetes`1.21`: `1.24` sono ora disponibili per i cluster locali su Outposts. AWS | Ora è possibile creare un cluster locale Amazon EKS su un Outpost utilizzando le versioni `1.21`-`1.24` di Kubernetes. In precedenza, era disponibile solo la versione `1.21`. | 17 gennaio 2023 |
| Amazon EKS ora supporta AWS PrivateLink | Puoi usare an AWS PrivateLink per creare una connessione privata tra il tuo VPC e Amazon EKS. | 16 dicembre 2022 |
| Supporto Windows per gruppi di nodi gestiti | Ora puoi utilizzare Windows per gruppi di nodi gestiti Amazon EKS. | 15 dicembre 2022 |
| I componenti aggiuntivi Amazon EKS di fornitori di software indipendenti sono ora disponibili nel Marketplace AWS | Ora puoi sfogliare e abbonarti ai componenti aggiuntivi Amazon EKS di fornitori di software indipendenti tramite il Marketplace. AWS | 28 novembre 2022 |
| AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 17 novembre 2022 |
| Kubernetes versione `1.24` | Sono stati aggiunti il supporto per Kubernetes versione `1.24` per nuovi cluster e aggiornamenti della versione. | 15 novembre 2022 |

| | | |
|---|---|---|
| Espansione AWS della regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Medio Oriente (Emirati Arabi Uniti) (`me-central-1` ). | 3 novembre 2022 |
| AWS aggiornamenti gestiti delle policy - Aggiornamento di una policy esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 24 ottobre 2022 |
| AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 20 ottobre 2022 |
| I cluster locali su AWS Outposts sono ora disponibili | Ora puoi creare un cluster locale Amazon EKS su un Outpost. | 19 settembre 2022 |
| Quote basate su vCPU Fargate | Fargate sta passando da quote basate su pod a quote basate su vCPU. | 8 settembre 2022 |
| AWS aggiornamenti gestiti delle politiche - Aggiornamento di una politica esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 31 agosto 2022 |
| Monitoraggio dei costi | Ora Amazon EKS supporta Kubecost, che consente di monitorare i costi suddivisi per risorse Kubernetes compresi pod, nodi, namespace ed etichette. | 24 agosto 2022 |
| AWS aggiornamenti gestiti delle politiche - Nuova politica | Amazon EKS ha aggiunto una nuova policy AWS gestita. | 24 agosto 2022 |
| AWS aggiornamenti gestiti delle politiche - Nuova politica | Amazon EKS ha aggiunto una nuova policy AWS gestita. | 23 agosto 2022 |

| | | |
|---|---|---|
| Aggiunta di tag alle risorse per la fatturazione | Aggiunto il supporto `aws:eks:cluster-name` generato per i tag di allocazione dei costi per tutti i cluster. | 16 agosto 2022 |
| Caratteri jolly del profilo Fargate | È stato aggiunto il supporto per i caratteri jolly del profilo Fargate nei criteri di selezione per i namespace, le chiavi e i valori delle etichette. | 16 agosto 2022 |
| Kubernetes versione `1.23` | Sono stati aggiunti il supporto per Kubernetes versione `1.23` per nuovi cluster e aggiornamenti della versione. | 11 agosto 2022 |
| Visualizza le risorse Kubernetes nel Console di gestione AWS | Ora è possibile visualizzare le informazioni relative alle risorse Kubernetes implementate nel cluster utilizzando la Console di gestione AWS. | 3 maggio 2022 |
| Espansione della regione Amazon EKS AWS | Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Jakarta) (`ap-southeast-3` ). | 2 maggio 2022 |
| Pagina Osservabilità e supporto per il componente aggiuntivo ADOT | Sono state aggiunte la pagina Observability e la AWS Distro for OpenTelemetry (ADOT). | 21 aprile 2022 |
| Kubernetes versione `1.22` | Sono stati aggiunti il supporto per Kubernetes versione `1.22` per nuovi cluster e aggiornamenti della versione. | 4 aprile 2022 |
| AWS aggiornamenti gestiti delle politiche - Nuova politica | Amazon EKS ha aggiunto una nuova policy AWS gestita. | 4 aprile 2022 |

| | | |
|---|---|---|
| Aggiunti i dettagli relativi all'applicazione di patch ai pod Fargate | Durante l'aggiornamento dei pod Fargate, Amazon EKS tenta innanzitutto di rimuovere i pod in base ai relativi budget di interruzione. È possibile creare regole di eventi per contrastare le espulsioni non riuscite prima che i pod vengano eliminati. | 1 aprile 2022 |
| Versione completa: Add-on supporto per il driver Amazon EBS CSI | Ora puoi utilizzare l' Console di gestione AWS interfaccia a riga di AWS comando e l'API per gestire il driver Amazon EBS CSI. | 31 marzo 2022 |
| AWS Aggiornamento dei contenuti di Outposts | Istruzioni per distribuire un cluster Amazon EKS su AWS Outposts. | 22 marzo 2022 |
| AWS aggiornamenti gestiti delle policy: aggiornamento a una policy esistente | Amazon EKS ha aggiornat o una policy AWS gestita esistente. | 21 marzo 2022 |
| Supporto `containerd` Windows | Ora puoi selezionare il runtime `containerd` per i nodi Windows. | 14 marzo 2022 |
| Sono state aggiunte alcune considerazioni in merito ad Amazon EKS Connector alla documentazione di sicurezza | Descrive il modello di responsabilità condivisa in relazione ai cluster connessi. | 25 febbraio 2022 |
| Assegnazione di indirizzi `IPv6` a pod e servizi | Ora è possibile creare un cluster `1.21` o successivo che assegna gli indirizzi `IPv6` a pod e servizi. | 6 gennaio 2022 |

| | | |
|---|---|---|
| AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente | Amazon EKS ha aggiornato una policy AWS gestita esistente. | 13 dicembre 2021 |
| Versione di anteprima: Add-on supporto per il driver Amazon EBS CSI | Ora puoi visualizzare l'anteprima utilizzando l'interfaccia a Console di gestione AWS riga di AWS comando e l'API per gestire il driver Amazon EBS CSI. | 9 dicembre 2021 |
| Supporto autoscaler Karpenter | Ora puoi utilizzare il progetto open source Karpenter per scalare automaticamente i nodi. | 29 novembre 2021 |
| Supporto filtro Fluent Bit Kubernetes nella registrazione di Fargate | Ora puoi usare il filtro Fluent Bit Kubernetes con la registrazione di Fargate. | 10 novembre 2021 |
| Supporto Windows disponibile nel piano di controllo | Il supporto Windows è ora disponibile nel piano di controllo. Non è più necessario attivarlo nel piano dati. | 9 novembre 2021 |
| Bottlerocket aggiunto come tipo di AMI per i gruppi di nodi gestiti | In precedenza, Bottlerocket era disponibile solo come opzione di nodo autogestito. Ora è possibile configurarlo come gruppo di nodi gestiti, riducendo lo sforzo necessario per soddisfare i requisiti di conformità dei nodi. | 28 ottobre 2021 |

| | | |
|---|---|---|
| Supporto driver DL1 | Le AMI Amazon Linux personalizzate ora supportano i carichi di lavoro di deep learning per Amazon Linux 2. Questa abilitazione consente una configurazione generica on-premises o di base su cloud. | 25 ottobre 2021 |
| Supporto video VT1 | Le AMI personalizzate Amazon Linux ora supportano VT1 per alcune distribuzioni. Questa abilitazione pubblicizza i dispositivi Xilinx U30 sul cluster Amazon EKS. | 13 settembre 2021 |
| È ora disponibile Amazon EKS Connector | Puoi utilizzare Amazon EKS Connector per registrare e connettere qualsiasi cluster Kubernetes conforme AWS e visualizzarlo nella console Amazon EKS. | 8 settembre 2021 |
| È ora disponibile Amazon EKS Anywhere | Amazon EKS Anywhere è una nuova opzione di implementazione per Amazon EKS che consente di creare e gestire cluster Kubernetes on-premises. | 08 settembre 2021 |
| Driver Amazon FSx per ONTAP CSI NetApp | È stato aggiunto un argomento che riassume il driver Amazon FSx for NetApp ONTAP CSI e fornisce collegamenti ad altri riferimenti. | 2 settembre 2021 |

| | | |
|---|---|---|
| Ora i gruppi di nodi gestiti calcolano automaticamente il numero massimo di pod Amazon EKS per i nodi | I gruppi di nodi gestiti ora calcolano automaticamente il numero massimo di pod Amazon EKS per i nodi implementati senza un modello di avvio, o con un modello di avvio in cui non è stato specificato un ID AMI. | 30 agosto 2021 |
| Rimozione della gestione delle impostazioni dei componenti aggiuntivi da parte di Amazon EKS, senza rimuovere il software aggiuntivo Amazon EKS | Ora è possibile rimuovere un componente aggiuntivo Amazon EKS senza rimuovere il software aggiuntivo dal cluster. | 20 agosto 2021 |
| Creazione di pod multi-homed con Multus | È possibile aggiungere più interfacce di rete a un pod utilizzando Multus. | 2 agosto 2021 |
| Aggiunta di altri indirizzi IP ai nodi Linux Amazon EC2 | È ora possibile aggiungere significativamente più indirizzi IP ai nodi Linux Amazon EC2. Ciò significa che è possibile eseguire una maggiore densità di pod su ciascun nodo. | 27 luglio 2021 |

| | | |
|---|---|---|
| `containerd` bootstrap del runtime | Ora l'Amazon Machine Image (AMI) di Amazon Linux ottimizzata e accelerata per Amazon EKS contiene un flag di bootstrap per abilitare facoltativamente il runtime `containerd` in AMI ottimizzate per Amazon EKS e Bottlerocket. Questo flag è disponibile in tutte le versioni Kubernetes supportate dell'AMI. | 19 luglio 2021 |
| Kubernetes versione `1.21` | Aggiunto il supporto per la versione `1.21` di Kubernetes. | 19 luglio 2021 |
| É stato aggiunto l'argomento relativo alle policy gestite | Un elenco di tutte le policy IAM gestite Amazon EKS e le modifiche apportate loro dal 17 giugno 2021. | 17 giugno 2021 |
| Utilizzo dei gruppi di sicurezza per i pod con Fargate | Ora è possibile usare gruppi di sicurezza per i pod con Fargate, oltre a utilizzarli con i nodi Amazon EC2. | 1 giugno 2021 |
| Sono stati aggiunti i componenti aggiuntivi CoreDNS e `kube-proxy` Amazon EKS | Amazon EKS può ora aiutarti a gestire i componenti aggiuntivi Amazon EKS `kube-proxy` e CoreDNS per il cluster. | 19 maggio 2021 |
| Kubernetes versione `1.20` | Sono stati aggiunti il supporto per Kubernetes versione `1.20` per nuovi cluster e aggiornamenti della versione. | 18 maggio 2021 |

| | | |
|---|---|---|
| AWS Rilasciato il Load Balancer Controller `2.2.0` | È ora possibile utilizzare il AWS Load Balancer Controller per creare Elastic Load Balancer utilizzando istanze o destinazioni IP. | 14 maggio 2021 |
| Contaminazioni di nodi per i gruppi di nodi gestiti | Amazon EKS ora supporta l'aggiunta di nodi taint ai gruppi di nodi gestiti. | 11 maggio 2021 |
| Crittografia dei segreti per i cluster esistenti | Amazon EKS ora supporta l'aggiunta della crittografia dei segreti per i cluster esistenti. | 26 febbraio 2021 |
| Kubernetes versione `1.19` | Sono stati aggiunti il supporto per Kubernetes versione `1.19` per nuovi cluster e aggiornamenti della versione. | 16 febbraio 2021 |
| Amazon EKS ora supporta i provider di identità OpenID Connect (OIDC) come metodo per autenticare gli utenti nel cluster aggiornato alla versione `1.16` o successiva. | I provider di identità OIDC possono essere utilizzati con o in alternativa all' AWS Identity and Access Management (IAM). | 12 febbraio 2021 |
| Visualizza le risorse relative ai nodi e ai carichi di lavoro in Console di gestione AWS | Ora è possibile visualizzare i dettagli sui nodi gestiti, autogestiti e Fargate e sui carichi di lavoro Kubernetes distribuiti nella Console di gestione AWS. | 1 dicembre 2020 |
| Implementazione dei tipi di istanza spot in un gruppo di nodi gestiti | Ora puoi distribuire più tipi di Spot o On-Demand Instance in un gruppo di nodi gestiti. | 1 dicembre 2020 |

| | | |
|---|---|---|
| Ora Amazon EKS può gestire componenti aggiuntivi specifici per il tuo cluster | Puoi gestire i component i aggiuntivi in autonomia o consentire ad Amazon EKS di controllare l'avvio e la versione di un componente aggiuntivo tramite l'API Amazon EKS. | 1 dicembre 2020 |
| Condivisione di un ALB su più ingressi | Ora puoi condividere un AWS Application Load Balancer (ALB) su più ingressi Kubernetes. In passato, era necessario implementare un ALB separato per ogni ingresso. | 23 ottobre 2020 |
| Supporto destinazioni IP per NLB | È ora possibile implementare un Network Load Balancer con destinazioni IP. Ciò significa che è possibile utilizzare un NLB per bilanciare il carico di traffico di rete verso i pod Fargate e direttamente verso i pod in esecuzione sui nodi Amazon EC2. | 23 ottobre 2020 |
| Kubernetes versione `1.18` | Sono stati aggiunti il supporto per Kubernetes versione `1.18` per nuovi cluster e aggiornam enti della versione. | 13 ottobre 2020 |
| Specificare un blocco CIDR personalizzato per l'assegna zione dell'indirizzo IP del servizio Kubernetes. | È ora possibile specificare un blocco CIDR personalizzato a cui Kubernetes assegna gli indirizzi IP di servizio. | 29 settembre 2020 |

| | | |
|---|---|---|
| Assegnazione dei gruppi di sicurezza a singoli pod | Ora è possibile associare diversi gruppi di sicurezza ad alcuni dei singoli pod in esecuzione su molti tipi di istanza Amazon EC2. | 09 settembre 2020 |
| Distribuzione di Bottlerocket sui nodi | È ora possibile implementare i nodi che eseguono Bottlerocket. | 31 agosto 2020 |
| La possibilità di avviare i nodi Arm è generalmente disponibile | È possibile avviare i nodi Arm in gruppi di nodi gestiti e autogestiti. | 17 agosto 2020 |
| Modelli di avvio di gruppi di nodi gestiti e AMI personalizzate | È possibile implementare un gruppo di nodi gestito che utilizza un modello di avvio di Amazon EC2. Se lo si preferisce, il modello di avvio può specificare un'AMI personalizzata. | 17 agosto 2020 |
| Supporto AWS EFS per Fargate | Ora puoi usare Amazon EFS con AWS Fargate. | 17 agosto 2020 |

| | | |
|---|---|---|
| Aggiornamento della versione della piattaforma Amazon EKS | Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. È incluso il supporto UDP per i servizi di tipo `LoadBalancer` quando si utilizzano Network Load Balancer con Kubernetes versione `1.15` o successiva. Per ulteriori informazioni, consulta il problema Consenti UDP for AWS Network Load Balancer su. GitHub | 12 agosto 2020 |
| Espansione della regione Amazon EKS AWS | Amazon EKS è ora disponibile nelle Regioni AWS Africa (Città del Capo) (`af-south-1`) ed Europa (Milano) (`eu-south-1`). | 6 agosto 2020 |
| Parametri di utilizzo Fargate | AWS Fargate fornisce metriche di CloudWatch utilizzo che forniscono visibilità sull'utilizzo delle risorse On-Demand Fargate da parte del tuo account. | 03 agosto 2020 |
| Kubernetes versione `1.17` | Sono stati aggiunti il supporto per Kubernetes versione `1.17` per nuovi cluster e aggiornamenti della versione. | 10 luglio 2020 |

| | | |
|---|---|---|
| Creazione e gestione delle risorse App Mesh da Kubernetes con il controller App Mesh per Kubernetes | Le risorse App Mesh possono essere create e gestite da Kubernetes. Inoltre, il controller inserisce automaticamente il proxy Envoy e i container init nei pod implementati. | 18 giugno 2020 |
| Amazon EKS ora supporta i nodi Inf1 Amazon EC2 | É possibile aggiungere nodi Inf1 Amazon EC2 al cluster. | 4 giugno 2020 |
| Espansione della regione Amazon EKS AWS | Amazon EKS è ora disponibile nelle AWS regioni AWS GovCloud (US-East) (`us-gov-east-1` ) e AWS GovCloud (US-Westus-gov-west-1 ) (). | 13 maggio 2020 |
| Kubernetes `1.12` non è più supportato in Amazon EKS | Kubernetes versione `1.12` non è più supportato in Amazon EKS. Esegui l'aggiornamento di eventuali cluster `1.12` alla versione `1.13` o successiva per evitare interruzioni del servizio. | 12 maggio 2020 |
| Kubernetes versione `1.16` | Sono stati aggiunti il supporto per Kubernetes versione `1.16` per nuovi cluster e aggiornamenti della versione. | 30 aprile 2020 |
| È stato aggiunto il ruolo AWSServiceRoleForAmazonEKS collegato al servizio | È stato aggiunto il ruolo collegato al AWSServiceRoleForAmazonEKS servizio. | 16 aprile 2020 |

| | | |
|---|---|---|
| Kubernetes versione 1.15 | Sono stati aggiunti il supporto per Kubernetes versione 1.15 per nuovi cluster e aggiornamenti della versione. | 10 marzo 2020 |
| Espansione della regione Amazon EKS AWS | Amazon EKS è ora disponibile nelle Regioni AWS Pechino (cn-north-1 ) e Ningxia (cn-northwest-1 ). | 26 febbraio 2020 |
| Driver CSI per Amazon FSx per Lustre | Aggiunto l'argomento relativo all'installazione del driver CSI per FSx per Lustre nei cluster Amazon EKS con Kubernetes 1.14. | 23 dicembre 2019 |
| Limitare l'accesso di rete all'endpoint di accesso pubblico di un cluster | Questo aggiornamento consente ad Amazon EKS di limitare gli intervalli CIDR che possono comunicare con l'endpoint di accesso pubblico del server API Kubernetes. | 20 dicembre 2019 |
| Risolvere l'indirizzo dell'endpoint di accesso privato per un cluster dall'esterno di un VPC | Con questo aggiornamento, è possibile utilizzare Amazon EKS per risolvere l'endpoint di accesso privato del server API Kubernetes dall'esterno di un VPC. | 13 dicembre 2019 |
| (Beta) Nodi di istanza Amazon EC2 A1 di Amazon EC2 | Avvio di nodi di istanza Amazon EC2 A1 di Amazon EC2 che si registrano con il cluster Amazon EKS. | 4 dicembre 2019 |
| Creazione di un cluster su AWS Outposts | Amazon EKS ora supporta la creazione di cluster su Outposts. AWS | 3 dicembre 2019 |

| | | |
|---|---|---|
| AWS Fargate su Amazon EKS | Ora i cluster Kubernetes Amazon EKS supportano i pod in esecuzione su Fargate. | 3 dicembre 2019 |
| Espansione della AWS regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Canada (Centraleca-central-1 ) (). | 21 novembre 2019 |
| Gruppi di nodi gestiti | I gruppi di nodi gestiti Amazon EKS automatizzano il provisioning la gestione del ciclo di vita dei nodi (istanze Amazon EC2) per i cluster Kubernetes Amazon EKS. | 18 novembre 2019 |
| Aggiornamento della versione della piattaforma Amazon EKS | Nuove versioni della piattaforma per gestire CVE-2019-11253. | 6 novembre 2019 |
| Kubernetes 1.11 non è più supportato in Amazon EKS | Kubernetes versione 1.11 non è più supportato in Amazon EKS. Esegui l'aggiornamento di eventuali cluster 1.11 alla versione 1.12 o successiva per evitare interruzioni del servizio. | 4 novembre 2019 |
| Espansione AWS della regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Sud America (San Paolosa-east-1 ) (). | 16 ottobre 2019 |
| Supporto Windows | Ora i cluster Amazon EKS che eseguono Kubernetes versione 1.14 ora supportano i carichi di lavoro di Windows. | 7 ottobre 2019 |

| | | |
|---|---|---|
| Scalabilità automatica | É stato aggiunto il capitolo per descrivere alcuni dei vari tipi di scalabilità automatic a Kubernetes supportati nei cluster Amazon EKS. | 30 settembre 2019 |
| Aggiornamento della dashboard Kubernetes | Aggiornato l'argomento relativo all'installazione della dashboard Kubernetes nei cluster Amazon EKS per l'uso della versione beta `2.0`. | 28 settembre 2019 |
| Driver CSI per Amazon EFS | Aggiunto l'argomento relativo all'installazione del driver CSI per Amazon EFS nei cluster Amazon EKS con Kubernetes `1.14`. | 19 settembre 2019 |
| Parametro di Amazon EC2 Systems Manager per ID di AMI ottimizzate per Amazon EKS | É stato aggiunto l'argomen to relativo al recupero dell'ID AMI ottimizzata per Amazon EKS mediante un parametro Amazon EC2 Systems Manager. Il parametro elimina la necessità di cercare gli ID AMI. | 18 settembre 2019 |
| Assegnazione di tag alle risorse di Amazon EKS | È possibile gestire l'assegna zione di tag ai propri cluster Amazon EKS. | 16 settembre 2019 |
| Driver CSI per Amazon EBS | Aggiunto l'argomento relativo all'installazione del driver CSI per Amazon EBS nei cluster Amazon EKS con Kubernetes `1.14`. | 9 settembre 2019 |

| | | |
|---|---|---|
| Nuova AMI ottimizzata per Amazon EKS con patch per `CVE-2019-9512` e `CVE-2019-9514` | Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per la gestione di CVE-2019-9512 e CVE-2019-9514. | 6 settembre 2019 |
| Annuncio della definizione di Kubernetes `1.11` come obsoleto in Amazon EKS | Amazon EKS ha interrotto il supporto per Kubernetes versione `1.11` il 4 novembre 2019. | 4 settembre 2019 |
| Kubernetes versione `1.14` | Sono stati aggiunti il supporto per Kubernetes versione `1.14` per nuovi cluster e aggiornamenti della versione. | 3 settembre 2019 |
| Ruoli IAM per gli account di servizio | Grazie ai ruoli IAM per gli account di servizio sui cluster Amazon EKS è possibile associare un ruolo IAM a un account di servizio Kubernetes. Grazie a questa funzionalità, non è più necessario fornire autorizzazioni estese al ruolo IAM del nodo. In questo modo, i pod su quel nodo possono chiamare le AWS API. | 3 settembre 2019 |
| Espansione AWS della regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Medio Oriente (Bahrain) (`me-south-1`). | 29 agosto 2019 |
| Aggiornamento della versione della piattaforma Amazon EKS | Nuove versioni della piattaforma per affrontare CVE-2019-9512 e CVE-2019-9514. | 28 agosto 2019 |

| | | |
|---|---|---|
| Aggiornamento della versione della piattaforma Amazon EKS | Nuove versioni della piattaforma da affrontare CVE-2019-11247 e CVE-2019-11249. | 5 agosto 2019 |
| Espansione delle Regioni di Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Hong Kongap-east-1 ) (). | 31 luglio 2019 |
| Kubernetes 1.10 non più supportato in Amazon EKS | Kubernetes versione 1.10 non è più supportato in Amazon EKS. Eseguire l'aggiornamento di ogni cluster 1.10 alla versione 1.11 o successiva per evitare interruzioni del servizio. | 30 luglio 2019 |
| Aggiunto argomento sul controller di ingresso ALB | L' AWS ALB Ingress Controller for Kubernetes è un controller che causa la creazione di un ALB quando vengono create le risorse di ingresso. | 11 luglio 2019 |
| Nuova AMI ottimizzata per Amazon EKS | Rimozione del file binario kubectl superfluo dalle AMI. | 3 luglio 2019 |
| Kubernetes versione 1.13 | Sono stati aggiunti il supporto per Kubernetes versione 1.13 per nuovi cluster e aggiornamenti della versione. | 18 giugno 2019 |
| Nuova AMI ottimizzata per Amazon EKS con patch per AWS-2019-005 | Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per risolvere le vulnerabilità descritte al link: security/ security -bulletins/ -2019-005 / [AWS-2019-005, type="marketing "].AWS | 17 giugno 2019 |

| | | |
|---|---|---|
| Annuncio dell'interruzione del supporto per Kubernetes `1.10` in Amazon EKS | Amazon EKS ha interrotto il supporto per Kubernete s versione `1.10` il 22 luglio 2019. | 21 maggio 2019 |
| Aggiornamento della versione della piattaforma Amazon EKS | Nuova versione della piattafor ma per cluster Kubernetes `1.11` e `1.10` per supportare nomi DNS personalizzati nel certificato `kubelet` e migliorar e le prestazioni di `etcd`. | 21 maggio 2019 |
| Nozioni di base su `eksctl` | Questa guida alle operazion i di base ti consente di creare tutte le risorse necessarie per iniziare a utilizzare `eksctl` in Amazon EKS. Questa è una semplice utility a riga di comando per la creazione e la gestione di cluster di Kubernetes su Amazon EKS. | 10 maggio 2019 |

| | | |
|---|---|---|
| AWS Comando CLI `get-token` | Il `aws eks get-token` comando è stato aggiunto alla AWS CLI. Non è più necessario installare AWS IAM Authenticator for Kubernetes per creare token di sicurezza client per la comunicazione con il server API del cluster. Aggiorna l'installazione della AWS CLI alla versione più recente per utilizzare questa nuova funzionalità. Per ulteriori informazioni, vedere Installazione dell'interfaccia a riga di AWS comando nella Guida per l'utente dell'interfaccia a riga di AWS comando. | 10 maggio 2019 |
| Aggiornamento della versione della piattaforma Amazon EKS | Nuova versione della piattaforma per cluster Kubernetes `1.12` per supportare nomi DNS personalizzati nel certificato `kubelet` e migliorare le prestazioni di `etcd`. La versione corregge un bug che costringeva i daemon `kubelet` dei nodi a richiedere un nuovo certificato a intervalli di pochi secondi. | 8 maggio 2019 |
| Tutorial su Prometheus | É stato aggiunto l'argomento relativo alla distribuzione di Prometheus nel cluster Amazon EKS. | 5 aprile 2019 |

| | | |
|---|---|---|
| Registrazione del piano di controllo Amazon EKS | Con questo aggiornamento, puoi ottenere registri di controllo e diagnostica direttamente dal piano di controllo di Amazon EKS. Puoi utilizzare questi CloudWatch log nel tuo account come riferimento per proteggere ed eseguire i cluster. | 4 aprile 2019 |
| Kubernetes versione `1.12` | Sono stati aggiunti il supporto per Kubernetes versione `1.12` per nuovi cluster e aggiornamenti della versione. | 28 marzo 2019 |
| É stata aggiunta la Guida alle operazioni di base di App Mesh | É stata aggiunta la documentazione per le nozioni di base su App Mesh e Kubernetes. | 27 marzo 2019 |
| Accesso privato all'endpoint del server API di Amazon EKS | Aggiunta la documentazione per la disabilitazione dell'accesso pubblico per l'endpoint del server API Kubernetes del cluster Amazon EKS. | 19 marzo 2019 |
| Aggiunto argomento per l'installazione del server dei parametri Kubernetes | Il server dei parametri Kubernetes è un aggregatore dei dati di utilizzo delle risorse nel cluster. | 18 marzo 2019 |
| Aggiunto elenco di progetti open source correlati | Questi progetti open source estendono le funzionalità dei cluster Kubernetes in esecuzione su AWS, inclusi i cluster gestiti da Amazon EKS. | 15 marzo 2019 |

| | | |
|---|---|---|
| Aggiunto argomento per l'installazione di Helm in locale | Il programma di gestione del pacchetto `helm` per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes. Questo argomento illustra come installare ed eseguire i binari `helm` e `tiller` localmente per consentire l'installazione e la gestione di grafici utilizzando la CLI helm nel sistema locale. | 11 marzo 2019 |
| Aggiornamento della versione della piattaforma Amazon EKS | Nuova versione della piattaforma che aggiorna i cluster Amazon EKS `1.11` Kubernetes a livello di patch da risolvere. `1.11.8` CVE-2019-1002100 | 8 marzo 2019 |
| Limite cluster aumentato | Amazon EKS ha aumentato il numero di cluster che puoi creare in una AWS regione da 3 a 50. | 13 febbraio 2019 |
| Espansione della AWS regione Amazon EKS | Amazon EKS è ora disponibile nelle AWS regioni Europa (Londra) (`eu-west-2`), Europa (Parigi) (`eu-west-3`) e Asia Pacifico (Mumbai) (`ap-south-1`). | 13 febbraio 2019 |
| Nuova AMI ottimizzata per Amazon EKS con patch per `ALAS-2019-1156` | Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per risolvere la vulnerabilità descritta in. ALAS-2019-1156 | 11 febbraio 2019 |

| | | |
|---|---|---|
| Nuova AMI ottimizzata per Amazon EKS con patch per `ALAS2-2019-1141` | Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per affrontare le problematiche di tipo CVE (Common Vulnerabilities & Exposures) riportate in ALAS2-2019-1141. | 9 gennaio 2019 |
| Espansione della AWS regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Seoul) (`ap-northeast-2`). | 9 gennaio 2019 |
| Espansione delle Regioni di Amazon EKS | Amazon EKS è ora disponibile nelle seguenti AWS regioni aggiuntive: Europa (Francoforte) (`eu-central-1`), Asia Pacifico (Tokyo) (`ap-northeast-1`), Asia Pacifico (Singapore) (`ap-southeast-1`) e Asia Pacifico (Sydney) (`ap-southeast-2`). | 19 dicembre 2018 |
| Aggiornamenti del cluster Amazon EKS | É stata aggiunta la documentazione relativa all'aggiornamento della versione del cluster Kubernetes e per la sostituzione dei nodi per Amazon EKS. | 12 dicembre 2018 |
| Espansione AWS della regione Amazon EKS | Amazon EKS è ora disponibile nella AWS regione Europa (Stoccolma) (`eu-north-1`). | 11 dicembre 2018 |

| | | |
|---|---|---|
| Aggiornamento della versione della piattaforma Amazon EKS | Nuova versione della piattaforma che aggiorna Kubernetes al livello di patch per `1.10.11` indirizzare il link: security/security -bulletins/AWS-2018-020/ [, type="marketing "]. CVE-2018-1002105 | 4 dicembre 2018 |
| È stato aggiunto il supporto alla versione `1.0.0` per il controller in ingresso ALB | Il controller di ingresso `1.0.0` ALB AWS rilascia una versione con supporto formale da. | 20 novembre 2018 |
| Aggiunto supporto per la configurazione di rete CNI | Ora il plugin CNI di Amazon VPC per Kubernetes versione `1.2.1` supporta la configurazione di rete personalizzata per interfacce di rete di pod secondari. | 16 ottobre 2018 |
| È stato aggiunto il supporto per `MutatingAdmissionWebhook` e `ValidatingAdmissionWebhook` | La versione della piattaforma Amazon EKS `1.10-eks.2` ora supporta i controller `MutatingAdmissionWebhook` e di ammissione `ValidatingAdmissionWebhook` . | 10 ottobre 2018 |
| Sono state aggiunte le informazioni dell'AMI del partner | Canonical ha collaborato con Amazon EKS per creare AMI del nodo utilizzabili nei cluster. | 3 ottobre 2018 |

| | | |
|---|---|---|
| Sono state aggiunte istruzioni per AWS il comando CLI `update-kubeconfig` | Amazon EKS ha aggiunto l'interfaccia `update-kubeconfig` a riga di AWS comando per semplificare il processo di creazione di un `kubeconfig` file per l'accesso al cluster. | 21 settembre 2018 |
| Nuove AMI ottimizzate per Amazon EKS | Amazon EKS ha aggiornato le AMI ottimizzate per Amazon EKS (con e senza il supporto di GPU) per fornire diverse correzioni di sicurezza e ottimizzazioni delle AMI. | 13 settembre 2018 |
| Espansione della AWS regione Amazon EKS | Amazon EKS è ora disponibile nella Regione UE (Irlanda) (`eu-west-1`). | 5 settembre 2018 |
| Aggiornamento della versione della piattaforma Amazon EKS | Nuova versione della piattaforma con supporto per il livello di aggregazione Kubernetes e per Horizontal Pod Autoscaler(HPA). | 31 agosto 2018 |
| Nuove AMI ottimizzate per Amazon EKS e supporto di GPU | Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per utilizzare un nuovo modello di AWS CloudFormation nodo e uno script di bootstrap. Inoltre, è disponibile una nuova AMI ottimizzata per Amazon EKS con supporto di GPU. | 22 agosto 2018 |

| | | |
|---|---|---|
| Nuova AMI ottimizzata per Amazon EKS con patch per ALAS2-2018-1058 | Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per affrontare le problematiche di tipo CVE (Common Vulnerabilities & Exposures) riportate in ALAS2-2018-1058. | 14 agosto 2018 |
| Script della build AMI ottimizzata per Amazon EKS | Amazon EKS ha reso open-source gli script di build che vengono utilizzati per creare l'AMI ottimizzata per Amazon EKS. Questi script di compilazione sono ora disponibili su. GitHub | 10 luglio 2018 |
| Versione iniziale di Amazon EKS | Documentazione iniziale per l'avvio del servizio | 5 giugno 2018 |

# Contribuire alla Guida per l'utente di EKS

AWS ha lanciato un'esperienza migliorata di contribuzione alla Guida per l'utente di EKS.

Ora è possibile modificare il codice sorgente della Guida per l'utente di EKS direttamente su GitHub.

I documenti ora utilizzano AsciiDoc, un potente linguaggio di authoring simile a markdown. AsciiDoc unisce una sintassi semplice a funzionalità di documentazione di livello aziendale come formattazione avanzata, riferimenti incrociati e controlli di sicurezza.

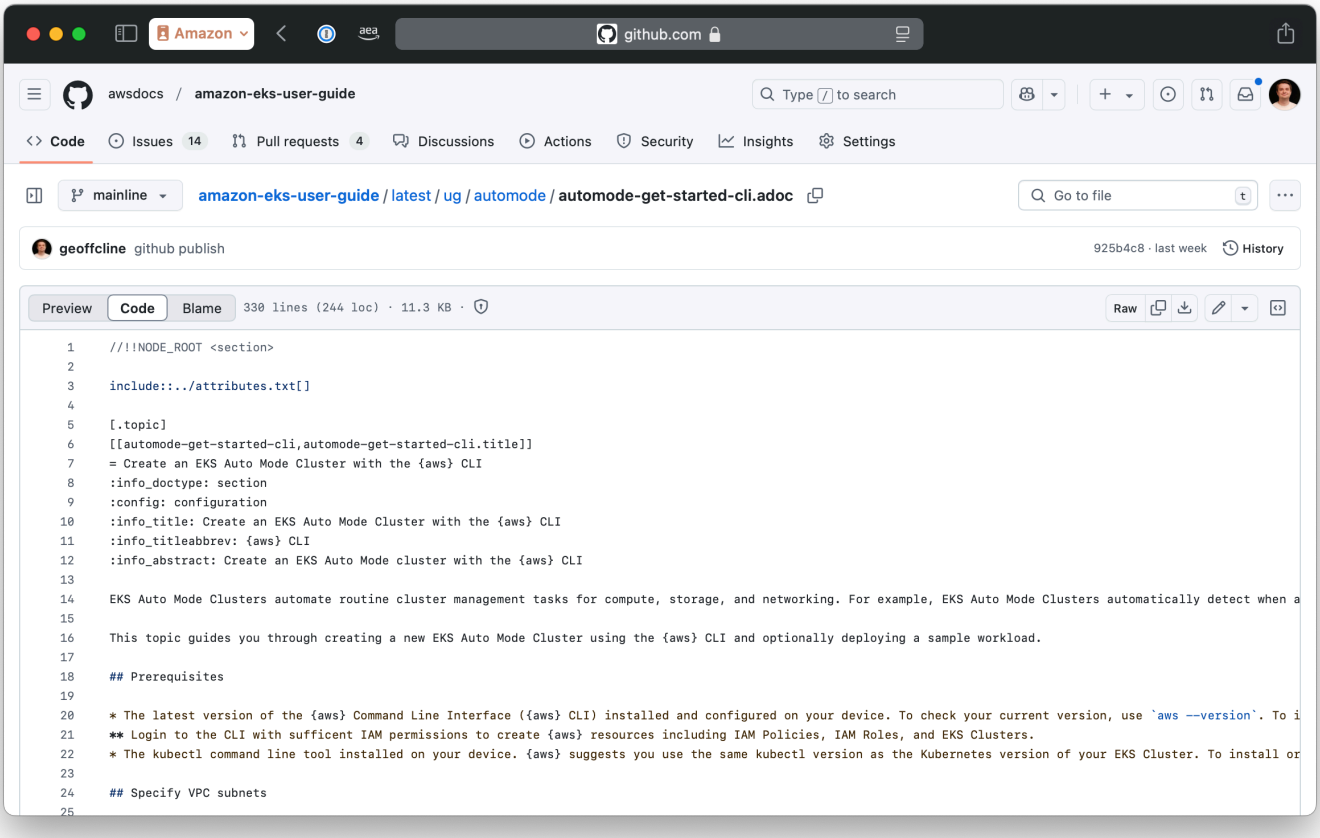
Ora è possibile modificare la documentazione EKS direttamente su GitHub. Il nostro processo ottimizzato include:

- Elaborazione più rapida delle richieste di pull.
- Fasi manuali ridotte.
- Controlli automatici della qualità dei contenuti.

Non vediamo l'ora di ricevere nuovi contributi.

## Modificare una singola pagina da un browser web

È possibile modificare una singola pagina della Guida per l'utente di EKS con facilità direttamente tramite browser web.




Se si desidera modificare più pagine da browser web, consultare la pagina the section called "Modifica i file con GitHub".

## Prerequisiti

- Pagina della documentazione da modificare aperta nel browser web.
- Effettuato l'accesso GitHub.

## Procedura

1. Accedere alla pagina da modificare nella Guida per l'utente di Amazon EKS.
2. Nel riquadro di destra, scegli il GitHub  link  Modifica questa pagina su.
3. Una volta acceso GitHub, apri l'editor in uno dei seguenti modi:
   - Premere il tasto e sulla tastiera.
   - Fare clic sull'icona a forma di matita e selezionare Modifica in posizione nel menu a discesa.

- Se non hai la possibilità di modificare, devi accedere a GitHub. Il tuo GitHub account non necessita di autorizzazioni speciali per suggerire modifiche. Tuttavia, i contributori interni di Amazon devono collegare il proprio profilo. GitHub

4. Apporta le modifiche richieste al contenuto dell'editor. GitHub
   - L'editor offre funzionalità di evidenziazione della sintassi e di anteprima.
   - Puoi usare il AsciiDoc markup per formattare le modifiche.
   - Puoi usarlo `ctrl-f` per aprire un' find/replace interfaccia.
5. (Facoltativo) Visualizzare le modifiche in anteprima.
   - Utilizzare la scheda `preview` per visualizzare un'anteprima delle modifiche con formattazione avanzata.
   - Utilizzare l'opzione `show diff` per evidenziare le sezioni modificate. Le sezioni rimosse presentano un indicatore rosso nel margine sinistro. Le nuove sezioni presentano un indicatore verde nel margine sinistro.
6. Al termine della modifica, fare clic sul pulsante Commit delle modifiche… nella parte alta dell'editor.
7. Nella finestra di dialogo di commit:
   - Verificare che l'indirizzo e-mail sia corretto.
   - Aggiungere un messaggio di commit breve ma descrittivo che spieghi le modifiche.
   - Facoltativamente, aggiungere una descrizione più lunga, se necessario.
   - Selezionare per creare una nuova ramificazione e richiesta di pull.

È stata creata una richiesta di pull che include le modifiche proposte.

## Panoramica della richiesta di pull

Quando si crea una richiesta pull:

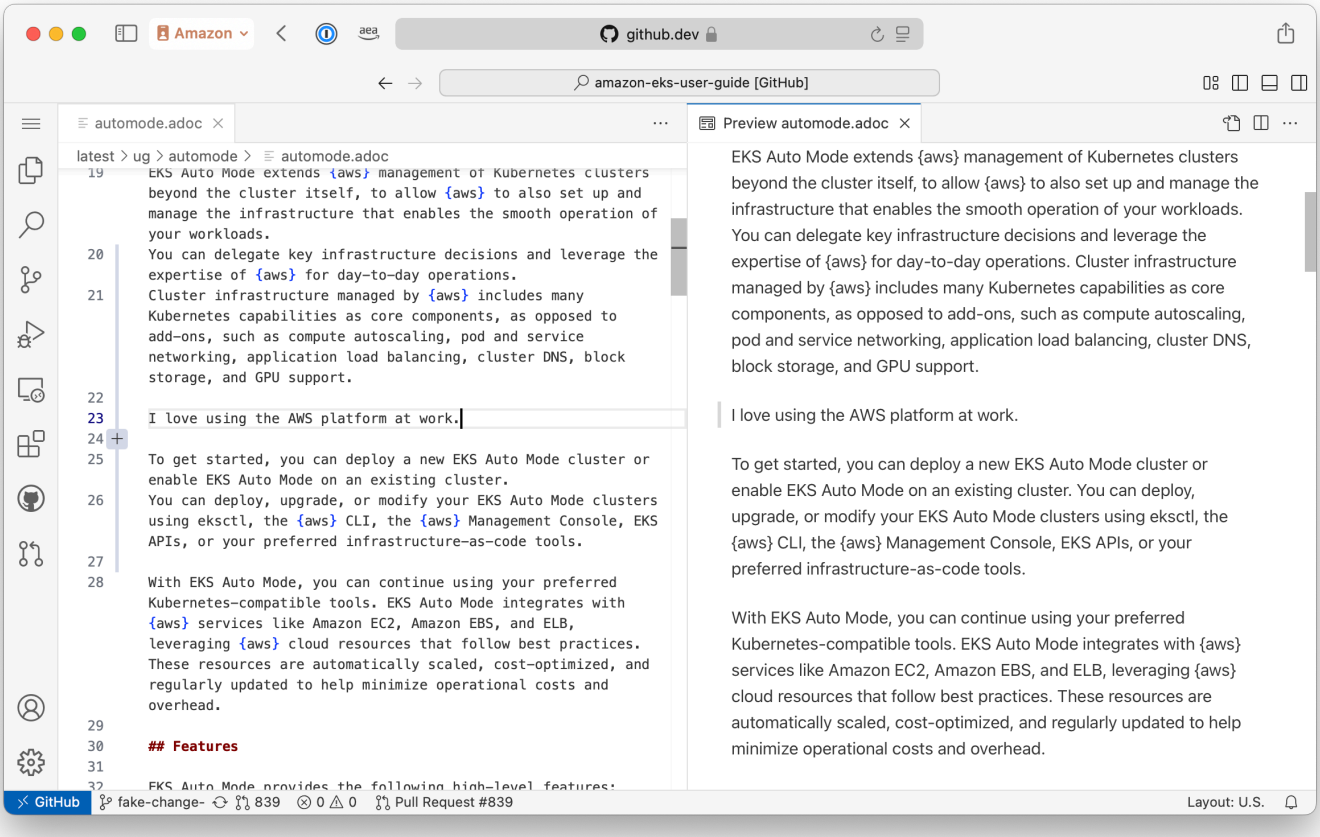
- Le modifiche vengono inviate e sottoposte a revisione da parte dei manutentori del repository.
- I revisori possono commentare i cambiamenti e richiedere modifiche.
- Per convalidare le modifiche potrebbero essere eseguiti test automatizzati.
- Una volta approvate, le modifiche possono essere unite al repository principale.

Le richieste di pull aiutano a garantire la qualità e offrono un modo per discutere le modifiche prima che vengano integrate.

[Scopri come le pull request vengono esaminate e approvate nei GitHub Documenti.]

## Modifica più file da un browser Web con GitHub Web Editor

Se desideri proporre modifiche a più pagine o creare una nuova pagina di documenti, usa l'editor GitHub.dev web. basato sull'editor di testo popolare Visual Studio Code.




## Prerequisiti

- Effettuato l'accesso a GitHub
- Familiarità con l'editor Visual Studio Code
- Familiarità con Git

# Procedura

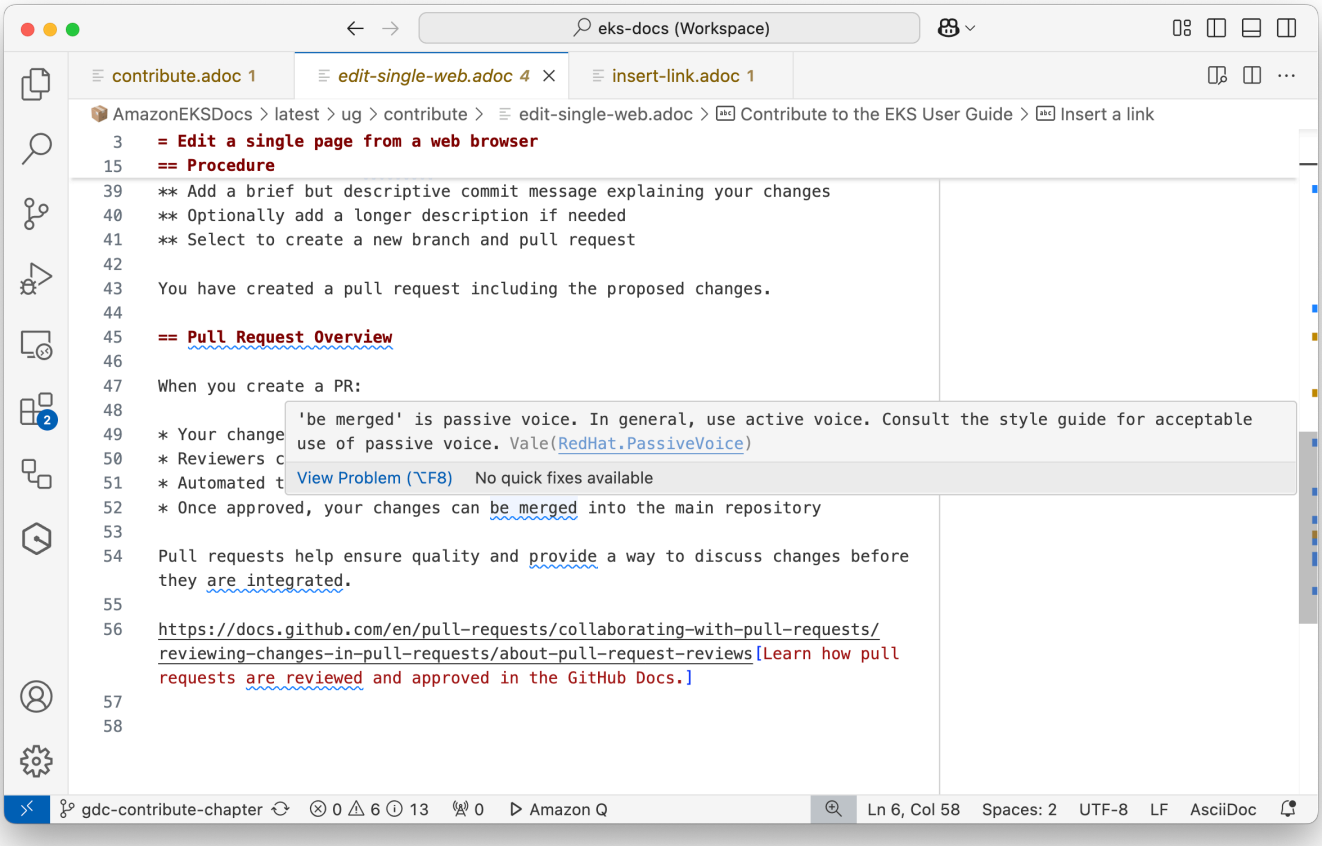
> **Note**
>
> Il team di EKS Docs ha creato un file di area di lavoro che include configurazioni suggerite per l'editor, come la disposizione del testo e l'evidenziazione della sintassi. AsciiDoc Suggeriamo di caricare questo file.

1. Apri l'area di lavoro su. GitHub.dev
   - È possibile aggiungere ai preferiti l'URL `https://github.dev/awsdocs/amazon-eks-user-guide/blob/mainline/eks-docs.code-workspace?workspace=true`.
2. (Solo per la prima configurazione) È possibile che ti venga richiesto di creare un fork del repository nel tuo account. GitHub Accettare questo prompt. Per ulteriori informazioni, consulta About forks nei documenti. GitHub
3. (Solo per la prima configurazione) Accetta la richiesta in basso a destra per installare l'estensione. AsciiDoc
4. Accedere ai contenuti della documentazione in `latest/ug`.
   - I file della documentazione sono organizzati per sezione principale. Ad esempio, i file di origine delle pagine del capitolo "Sicurezza" si troveranno nella directory "security/".
5. Per visualizzare un'anteprima della pagina di documentazione, utilizzare il pulsante Apri anteprima a lato in alto a destra. L'icona include una piccola lente d'ingrandimento.
6. Utilizzare la scheda Controllo del codice sorgente a sinistra per eseguire il commit delle modifiche. Per ulteriori informazioni, consultare le seguenti pagine della documentazione di Visual Studio Code:
   - Commit changes
   - Creazione di una richiesta di pull

Una volta creata una richiesta di pull, sarà revisionata dal team di documentazione.

# Visualizzazione di un feedback sullo stile durante la digitazione installando Vale localmente

È possibile vedere il feedback sullo stile durante la digitazione. Ciò aiuta a identificare la scrittura scorretta e gli errori di battitura.




Panoramica:

- CLI Vale carica le guide di stile e le avvia sui file sorgente.
- Il repository EKS Docs include un file di configurazione Vale che carica guide di stile e regole locali.
- L'estensione Vale per Visual Studio (VS) Code mostra il feedback sui valori all'interno dell'editor.

## Installa Java

Segui le istruzioni nei documenti della CLI Vale per installare Vale con un sistema di gestione pacchetti.

## Installa l'estensione VS Code Vale

1. Apri VS Code.
2. Fai clic sull'icona delle estensioni nella barra delle attività (o premi Ctrl+Shift+X).
3. Cerca "Vale".
4. Fai clic su Installa sull'estensione "Vale VSCode" di Chris Chinchilla.
5. Ricarica VS Code quando richiesto.

## Sincronizzazione di Vale

Vale utilizza il file di configurazione di `.vale.ini` nella root del progetto per determinare quali regole di stile applicare.

1. Apri VS Code.
2. Fai clic su Visualizza > Terminale (o premi Ctrl+`).
3. Se necessario, accedi alla directory principale del progetto.
4. Eseguire il comando :

```
vale sync
```

5. Attendi che Vale finisca di scaricare e sincronizzare le regole di stile.

## Visualizza il feedback di stile in VS Code

1. Apri un Markdown o un AsciiDoc file in VS Code.
2. L'estensione Vale controllerà automaticamente il testo in base alle regole di stile.
3. I problemi di stile saranno sottolineati nell'editor.
4. Passa il mouse sul testo sottolineato per visualizzare il suggerimento di stile specifico.
5. Risolvi i problemi seguendo i suggerimenti o consultando la guida di stile.

# Creazione di una nuova pagina

Scopri come creare una nuova pagina di documentazione. Questo argomento include le istruzioni per creare i metadati della pagina iniziale e aggiungere la pagina all'indice della guida.

## Creazione di una pagina

1. Passa alla directory del capitolo. Ad esempio, se vuoi creare una nuova pagina nella sezione "Sicurezza", vai alla directory `latest/ug/security`.
2. Determina l'ID della pagina. Per convenzione, l'ID della pagina è tutto minuscolo e segmentato con `-`. L'ID di questa pagina è `create-page`.
3. Crea un nuovo file con l'ID della pagina e l'estensione `adoc`. Ad esempio, `create-page.adoc`.
4. Inserisci i metadati della pagina usando questo modello:

```
include::../attributes.txt[]

[.topic]
[#unique-page-id]
= Page title
:info_titleabbrev: Short title

[abstract]
--
Brief summary of the page's content.
--

Introduction paragraph goes here.
```

## Aggiungi pagina alla navigazione

1. Passa alla pagina principale. La pagina principale delle sezioni di primo livello è `book.adoc`.
2. Nella parte inferiore della pagina principale, inserisci la pagina secondaria.

   `include::${filename}[leveloffset=+1]`

   Ad esempio:

   `include::create-page.adoc[leveloffset=+1]`

# Inserire un link

AsciiDoc supporta diversi tipi di link. L'utilizzo del tipo di collegamento corretto è importante affinché quest'ultimo funzioni correttamente in diversi ambienti.

# Collegamento a una pagina o sezione della Guida per l’utente EKS

Utilizza i riferimenti incrociati (xref) per collegare pagine o sezioni all’interno dello stesso sito di documentazione, come la Guida per l’utente EKS. Si aggiornano automaticamente se la sezione di destinazione viene spostata o rinominata.

## Usa il titolo della pagina come testo del link

Nella maggior parte dei casi, quando ti colleghi a un altro ID in questa guida per l’utente, utilizza il seguente approccio per fare in modo che il testo del link venga aggiornato automaticamente al titolo più recente, se necessario.

```
For more information, see <<page-or-section-id>>.
```

## Definisci il testo del collegamento personalizzato

Nei casi in cui è necessario disporre di un testo di collegamento personalizzato, utilizza il seguente formato.

```
Here's an example of a <<page-or-section-id,link with custom text>>.
```

# Collegamento a un'altra guida nei AWS Documenti

1. Trova il link alla pagina della AWS documentazione.
2. Rimuovi il prefisso `https://docs.aws.amazon.com/`, mantenendo solo il percorso. Il percorso deve iniziare con una lettera.
3. Crea un link come illustrato qui di seguito:

```
link:AmazonS3/latest/userguide/create-bucket-overview.html[Create a bucket,
 type="documentation"]
```

# Collegamento a una pagina Web esterna

Questo formato crea un collegamento standard a una pagina non ospitata da Amazon. Ad esempio, usalo per GitHub i link.

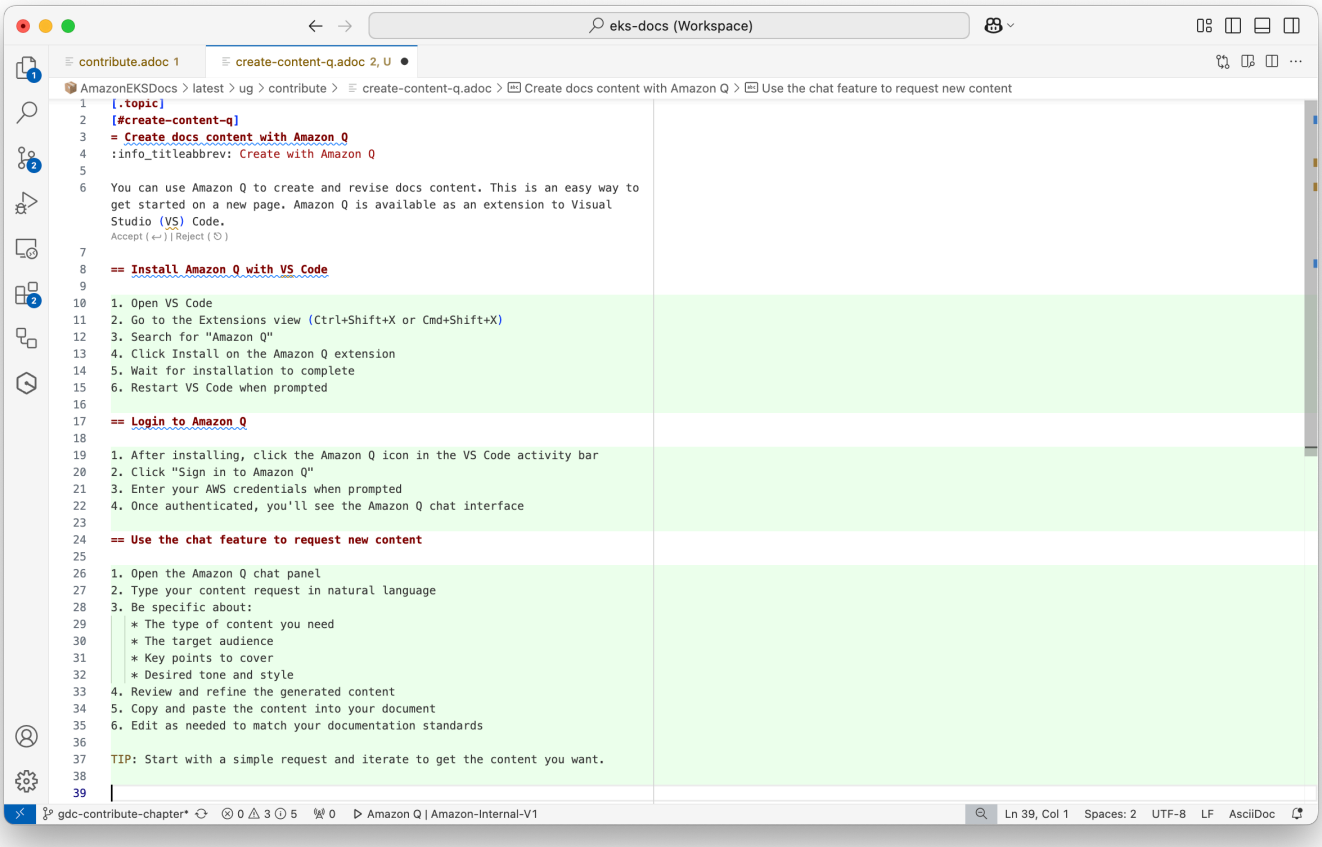
```
https://example.com[Link text]
```

> **Note**
>
> Abbiamo una lista consentita per i domini esterni. La lista consentita si trova in `vale/styles/EksDocs/ExternalDomains.yml`

# Creare contenuti per documenti con Amazon Q

È possibile utilizzare Amazon Q per creare e modificare il contenuto dei documenti. Si tratta di un modo semplice per iniziare a scrivere una nuova pagina. Amazon Q è disponibile come estensione in Visual Studio (VS) Code.

Nell'immagine seguente, Amazon Q ha generato le righe evidenziate in verde.




# Installare Amazon Q con VS Code

1. Apri VS Code.

2. Vai alla vista Estensioni (Ctrl+Shift+X o Cmd+Shift+X).
3. Cerca «Amazon Q».
4. Scegli Installa sull'estensione Amazon Q.
5. Attendi il completamento dell'installazione.
6. Riavvia VS Code quando richiesto.

## Accedi ad Amazon Q

1. Dopo l'installazione, scegliere l'icona Amazon Q nella barra di attività di VS Code.
2. Scegliere Accedi ad Amazon Q.
3. Inserisci AWS le tue credenziali quando richiesto.
4. Dopo l'autenticazione, viene visualizzata l'interfaccia di chat di Amazon Q.

## Utilizzare Amazon Q per creare i contenuti

1. Aprire il file da modificare in VS Code.
2. Selezionare il testo da revisionare o la posizione per i nuovi contenuti.
3. Premere Ctrl+I o Cmd+I.
4. Nel prompt, specificare con precisione:
   - Il tipo di contenuto necessario.
   - I destinatari.
   - I punti principali da trattare.
   - Registro e stile desiderati.
5. Revisionare il contenuto generato nell'anteprima in linea.
6. Utilizzare invio per accettare le modifiche o esc per rifiutarle.
7. Apportare ulteriori modifiche se necessario.

## Suggerimenti

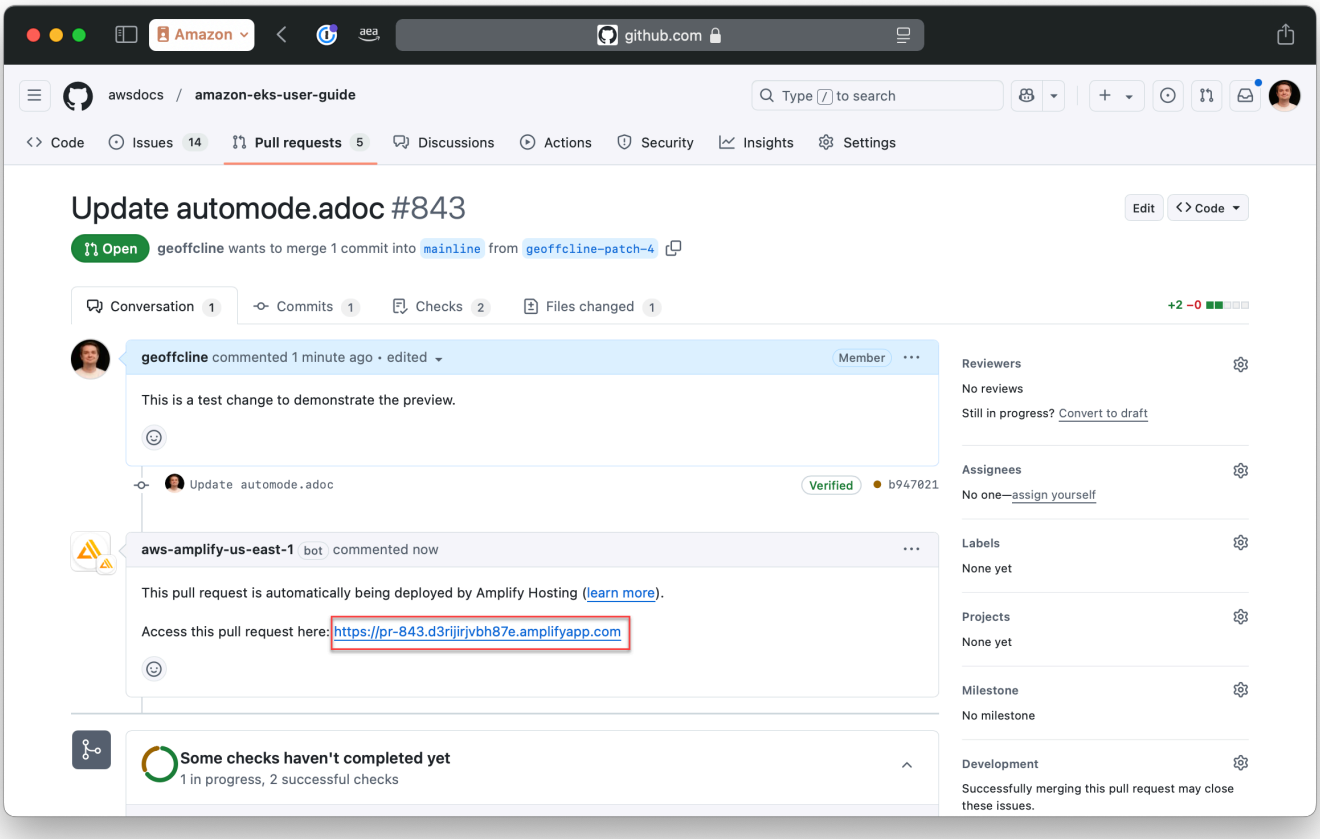
- Iniziare da una richiesta semplice e iterare per ottenere il contenuto desiderato.
- Creare una prima bozza dei titoli delle pagine, poi chiedere a Q di completarli.

- Amazon Q potrebbe generare output in Markdown. Questo linguaggio è accettabile. Gli AsciiDoc strumenti sono in grado di comprendere la maggior parte della sintassi di markdown.

Per ulteriori informazioni su Amazon Q Developer, consultare la pagina Utilizzo di Amazon Q Developer nell'IDE.

# Visualizzazione di un'anteprima del contenuto della richiesta di pull

La Amazon EKS User Guide GitHub è configurata per creare e generare un'anteprima del sito di documenti. Questa anteprima non include il AWS tema completo, ma verifica che il contenuto sia stato creato correttamente e i collegamenti funzionino.




Questa anteprima è ospitata su un URL temporaneo da AWS Amplify.

## Visualizzazione dell'anteprima

Quando invii una pull request, AWS Amplify tenta di creare e distribuire un'anteprima del contenuto.

Se la compilazione ha esito positivo, aws-amplify-us-east-1 aggiunge un commento alla richiesta di pull con un collegamento all'anteprima. Scegli il link a destra di "Accedi a questa richiesta di pull qui" (come indicato nello screenshot con un contorno rosso).

Se la compilazione ha esito negativo, gli amministratori del repository possono vedere i log e fornire feedback.

> **Note**
>
> Se non hai mai contribuito prima, potrebbe essere necessario che un responsabile del progetto approvi l'esecuzione della build.

## Limitazioni dell'anteprima

L'anteprima è creata come un unico file HTML di grandi dimensioni. Una volta pubblicato, sarà visualizzato come più pagine.

Cosa funziona:

- Riferimenti incrociati (`xref`)
- Link a Internet
- Immagini
- Contenuti ospitati da `samples/`

Cosa non funziona:

- Collegamenti ad altri AWS contenuti, utilizzando. `type="documentation"` Ciò accade perché questo contenuto non esiste nell'ambiente di anteprima.
- L'attributo `{aws}` non sarà visualizzato correttamente. Il suo valore cambia in base all'ambiente.

# AsciiDoc riferimento alla sintassi

AsciiDoc è il linguaggio di markup preferito per la AWS documentazione. Sebbene gli strumenti supportino parzialmente la sintassi di Markdown (come titoli ed elenchi), consigliamo di utilizzare la AsciiDoc sintassi per tutti i contenuti per garantire la coerenza e la corretta visualizzazione.

Questa guida illustra gli elementi di sintassi più comuni necessari per contribuire alla documentazione di Amazon EKS. Per una sintassi più avanzata e informazioni dettagliate, consulta la documentazione. AsciiDoc

## Nuova pagina

Ogni nuovo AsciiDoc documento dovrebbe iniziare con la struttura definita inthe section called "Creazione di una pagina".

## Intestazioni

Usa le intestazioni per organizzare i contenuti in modo gerarchico:

```
= Page/topic title (level 1)
== Section title (level 2)
=== Level 3 heading
==== Level 4 heading
===== Level 5 heading
====== Level 6 heading
```

Nota: usa sempre le lettere maiuscole e minuscole per i titoli della AWS documentazione.

## Formattazione del testo

Formattare il testo per enfatizzare le informazioni importanti:

```
Use *bold text* for UI labels.
Use _italic text_ for introducing terms or light emphasis.
Use `monospace text` for code, file names, and commands.
```

## Elenchi

### Elenchi non ordinati

Creare elenchi puntati per gli elementi senza una sequenza specifica:

```
* First item
* Second item
** Nested item
```

```
** Another nested item
* Third item
```

### Elenchi ordinati

Creare elenchi numerati per fasi sequenziali o elementi con priorità:

```
. First step
. Second step
.. Substep 1
.. Substep 2
. Third step
```

## Link

Vedi the section called "Inserire link" i dettagli su come formattare correttamente i link. Markdown-style i collegamenti non sono supportati.

## Esempi di codice

### Codice inline

Usare gli apici retroversi per il codice inline:

```
Use the `kubectl get pods` command to list all pods.
```

### Blocchi di codice

Creare blocchi di codice con evidenziazione della sintassi e supporto per gli attributi (simili alle entità):

```
[source,python,subs="verbatim,attributes"]
----
def hello_world():
    print("Hello, World!")
----
```

## Immagini

Inserire immagini con testo alternativo per l'accessibilità:

```
image::images/image-file.png[Alt text description]
```

## Tabelle

Creare tabelle per organizzare le informazioni:

```
[%header,cols="2"]
|===
|Header 1
|Header 2

|Cell 1,1
|Cell 1,2

|Cell 2,1
|Cell 2,2
|===
```

Per tabelle più complesse, consulta la documentazione della AsciiDoc tabella.

## Callout

Usare i callout per evidenziare informazioni e ammonizioni importanti:

```
NOTE: This is a note callout for general information.

TIP: This is a tip callout for helpful advice.

IMPORTANT: This is an important callout for critical information.
```

Anteprima:

> **Note**
>
> Questo è un callout di nota.

## Includere altri file

Includere contenuti da altri file:

```
include::filename.adoc[]
```

## Attributi (simili alle entità)

Utilizzare attributi predefiniti per mantenere la coerenza. In particolare, è NECESSARIO utilizzare gli attributi per AWS e `arn:aws:` .

```
{aws} provides Amazon EKS as a managed Kubernetes service.
```

```
[source,bash,subs="verbatim,attributes"]
----
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn {arn-aws}iam::aws:policy/AmazonEKSClusterPolicy
----
```

Per un elenco degli attributi, cercare nel file `../attributes.txt`.

## Procedure

Procedure dettagliate per il formato:

```
To create an Amazon EKS cluster, do the following steps.

. Sign in to the {aws} Management Console.
. Open the Amazon EKS console.
. Choose *Create cluster*.

...
```